МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему: «Розробка рекомендаційної системи ігрової платформи з використанням графових баз даних»

здобувача Щіпки Олексія Романовича _____
(ПІБ) *(Підпис)*

Науковий керівник:
проф., д.т.н., с.н.с.
_____Артемчук В.О.

Робота допущена до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент
_____Тішков Б.О.

Київ 2025

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:
Керівник проектної групи(гарант)
освітньо-професійної програми
_____Помазун О.М.
"____" _____2025 р.

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____Тішков Б.О.
"____" _____2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувачу вищої освіти ***Щіпки Олексія Романовича***

очної (денної) форми навчання

на підготовку кваліфікаційної бакалаврської роботи
на тему: «Розробка рекомендаційної системи ігрової платформи з використанням графових баз даних»

Тему затверджено наказом ректора Університету від « *7* » *березня 2025* р. № *466- ст*.
Кваліфікаційна бакалаврська робота виконується на матеріалах проектування та розробки рекомендаційної системи в рамках ігрової платформи з використанням графових баз даних та сучасних технологій обробки даних.

План кваліфікаційної бакалаврської роботи

Розділ І Аналіз та установлення вимог до рекомендаційної системи платформи ігрової дистрибуції

Розділ ІІ Розроблення програмного та алгоритмічного забезпечення рекомендаційної системи

Розділ ІІІ Проєктування та реалізація системних компонентів

Об'єкт дослідження Процес створення рекомендаційної системи, реалізованої на поєднані графової бази даних із реляційною, що забезпечує високу швидкодію та простоту у заповнені і виведенні даних.

Предмет дослідження Методи та інструменти побудови рекомендаційних систем з використанням графових баз даних для аналізу зв'язків між іграми, жанрами, сеттінгами та тегами.

Мета кваліфікаційної бакалаврської роботи Створити прототип рекомендаційної системи, яка поєднує переваги графових і реляційних БД, з використанням модуля psycopg2 для інтеграції та запиту даних із PostgreSQL для персоналізованого підбору ігор.

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I Провести аналіз актуальності проблеми персоналізованих рекомендацій та обґрунтувати необхідність розробки нової системи. Виконати огляд та аналіз існуючих аналогів платформ та рекомендаційних систем для дистрибуції ігор, виявити їхні переваги та недоліки. Сформулювати цілі та завдання розробки рекомендаційної системи для отримання персоналізованих рекомендацій. Деталізувати функціональні вимоги до системи (запити на гру, індивідуальний пошук, автентифікація, журнал відслідковуваних ігор). Визначити нефункціональні вимоги до системи (безпека, продуктивність, надійність, масштабованість, зручність використання).

У розділі II Обґрунтувати вибір технологічного стеку (Neo4j, PostgreSQL, Python, FastAPI з REST API) для розробки системи. Розробити загальну архітектуру інформаційної системи. Спроектувати та описати основні функціональні модулі бекенду (бази даних ігор, тегів та формування запитів). Спроектувати та описати інтерфейси користувача та сценарії взаємодії із застосунком. Визначити принципи реалізації основних компонентів системи та їхню взаємодію.

У розділі III Розробити та описати інформаційне забезпечення системи, включаючи інфологічну та даталогічну моделі бази даних. Обґрунтувати поділ даних між графовою та реляційною структурами, визначено механізми зберігання, обробки та фільтрації. Також розглянути технічне забезпечення: параметри серверного хостингу, апаратного середовища для розгортання API та баз даних, вимоги до клієнтської сторони. Визначени склад системного та прикладного програмного забезпечення, а також зміст документації, необхідної для підтримки проєкту. Провести пілотну використання— перевірку працездатності системи на тестовому наборі ігор і користувачів. Проаналізувати результати реалізації, оцінити цінність практичну та сформулювати подальші рекомендації для вдосконалення.

Завдання підготував
науковий керівник _____ **Артемчук Володимир Олександрович**
«10 » березня 2025 р.

Завдання одержав
здобувач _____ **Щіпка Олексій Романович**
«10 » березня 2025 р.

Відгук
про кваліфікаційну бакалаврську роботу
здобувача навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми
«Комп'ютерні науки»

на тему
«Проєктування інформаційної системи для захищеного завантаження файлів»

1. Актуальність теми: Тема кваліфікаційної бакалаврської роботи є надзвичайно актуальною у зв'язку зі стрімким розвитком ігрової індустрії та необхідністю покращення механізмів взаємодії між користувачем і контентом. В умовах, коли кількість ігор на ринку щороку зростає в геометричній прогресії, ефективна система рекомендацій є важливим інструментом як для користувачів, так і для розробників. Стандартні підходи до рекомендацій мають обмеження у гнучкості, масштабованості та якості персоналізації. Використання графових баз даних (Neo4j) у поєднанні з реляційною (PostgreSQL) дає змогу моделювати складні взаємозв'язки між жанрами, сеттінгами, тегами та користувачами, що робить тему дослідження актуальною й практично значущою.

2. Позитивні риси кваліфікаційної бакалаврської роботи: Робота вирізняється системністю підходу та сучасним стеком технологій: FastAPI, Python, pycopg2, PostgreSQL, Neo4j, React. Здобувач реалізував клієнт-серверну архітектуру, графову модель для взаємозв'язків між іграми та жанрами, а також SQL-модель для зберігання тегів. Значну увагу приділено побудові REST API, реалізації запитів мовою Cypher і інтеграції даних між базами. Варто відзначити також наявність вебінтерфейсу, який відображає результати роботи системи та забезпечує взаємодію з користувачем у зручній формі.

3. Наявність самостійних розробок автора: Здобувач продемонстрував високий рівень самостійності під час реалізації проєкту. Ним було самостійно спроектовано структуру графової та реляційної моделей, реалізовано обмін даними між базами за допомогою бібліотеки pycopg2, налаштовано FastAPI-сервер для обробки запитів і створено інтерфейс користувача з використанням React. Особистий внесок автора простежується також у побудові алгоритмів формування рекомендацій та структуризації запитів користувача.

4. Цінність теоретичних висновків та практичних рекомендацій: Теоретична цінність роботи полягає у розробці моделі, яка поєднує переваги графових та реляційних БД у системах рекомендацій. Практичний результат — функціонуючий прототип, який можна масштабувати та адаптувати для реального використання на

ігрових платформах. Надані рекомендації стосуються впровадження додаткової логіки обліку поведінки користувача, автоматичного навчання на основі активності та реалізації системи зворотного зв’язку.

5. Наявність недоліків: Серед недоліків слід зазначити відсутність інтеграції з реальними платформами дистрибуції ігор (наприклад, Steam API) та відсутність модулів машинного навчання, що обмежує персоналізацію на поведінковому рівні. Також не було проведено повноцінного навантажувального тестування, що могло б виявити вузькі місця в архітектурі. Проте ці обмеження є природними для обсягу бакалаврської роботи.

6. Загальна оцінка кваліфікаційної бакалаврської роботи та її допущення до захисту перед ЕК: Кваліфікаційна бакалаврська робота на тему «Розробка рекомендаційної системи ігрової платформи з використанням графових баз даних» є завершеним, логічно побудованим дослідженням, яке повністю відповідає поставленим завданням. Вона демонструє глибокі знання здобувача в галузі веброзробки, баз даних, програмної інженерії та інтелектуального аналізу даних. Робота має практичну цінність та заслуговує на високу оцінку. Рекомендується до захисту перед Екзаменаційною комісією.

Науковий керівник

проф., д.т.н., с.н.с. Артемчук В.О.

_____
(підпис)

“__”_____ 20__ р.

Рецензія
на кваліфікаційну бакалаврську роботу
здобувача вищої освіти

_____

(прізвище, ім'я, по батькові)

Тема «Розробка рекомендаційної системи ігрової платформи з використанням графових баз даних»

Актуальність теми кваліфікаційної бакалаврської роботи і доцільність її розроблення: Тема є актуальною в умовах стрімкого розвитку ігрової індустрії та зростаючого обсягу цифрового контенту. В умовах високої конкуренції та інформаційного перевантаження користувачі потребують інтелектуальних інструментів, які допомагають орієнтуватися у великій кількості ігор. Створення рекомендаційної системи, що використовує графові бази даних для аналізу жанрів, сеттінгів і тегів, відповідає сучасним викликам у сфері цифрової дистрибуції. Застосування технологій Neo4j, PostgreSQL та FastAPI забезпечує гнучкість, масштабованість і пояснюваність рекомендацій.

Якість проведеного дослідження: Робота виконана на високому рівні та містить систематизований аналіз предметної галузі, обґрунтування вибору архітектури, побудову моделей даних та реалізацію основного функціоналу системи. Здобувач продемонстрував впродемонстрував знання у сфері побудови графових структур, взаємодії між базами даних за допомогою psycopg2, побудови REST API та створення інтерфейсу з використанням React. Особливу увагу приділено логіці формування персоналізованих рекомендацій на основі складних зв'язків між іграми.

Позитивні риси кваліфікаційної бакалаврської роботи: До позитивних рис роботи слід віднести використання сучасного та функціонального технологічного стеку (React/TypeScript для фронтенду, Node.js/Express для бекенду, MongoDB як база даних), що відображає актуальні тенденції в галузі. Особливою перевагою є комплексний підхід до реалізації безпекових механізмів, що включає хешування паролів, використання JWT для автентифікації, реалізацію одноразових паролів з обмеженням спроб та систему детального логування доступу. Робота демонструє вміння здобувача не тільки проектувати, а й втілювати функціональні та захищені програмні продукти. Інтерфейс системи є інтуїтивно зрозумілим, що сприяє зручності користувача.

Зауваження: Попри високий рівень виконання роботи, як і в будь-якому проекті в рамках бакалаврської кваліфікації, можна відзначити кілька зауважень, які не

применшують її загальної цінності. Реалізований прототип, хоча і є повноцінним, не охоплює всіх можливих сценаріїв для повномасштабного комерційного використання, наприклад, розширені ролі та права доступу користувачів чи інтеграція з корпоративними системами автентифікації. Детальне тестування продуктивності системи під екстремальним навантаженням та формальний зовнішній аудит безпеки (penetration testing) не були предметом даного дослідження, що, втім, є типовим для робіт такого рівня.

Практична значимість висновків і рекомендацій: Отримані результати мають високу практичну цінність для реалізації систем персоналізованого підбору ігор на платформах цифрової дистрибуції. Запропонована гібридна архітектура (Neo4j + PostgreSQL) дозволяє реалізувати масштабовану, ефективну та прозору систему рекомендацій, яка може бути інтегрована у комерційні сервіси або адаптована під академічні й навчальні проєкти. Описані у роботі технологічні та методологічні підходи можуть бути використані як основа для подальших досліджень у сфері інтелектуальних інформаційних систем.

Місце роботи та посада рецензента
Науковий ступінь, учене звання (*за наявності*) _____
(підпис, ПІБ)

Підпис засвідчую: _____
(посада, підпис)

Місце печатки організації, де працює рецензент

АНОТАЦІЯ

кваліфікаційної бакалаврської роботи
здобувача першого (бакалаврського) рівня вищої освіти, 4 курсу,
виконаної на тему: «Розробка рекомендаційної системи ігрової платформи з використанням графових баз даних»
Київ: кафедра інформаційних систем в економіці, 2025 р.

Кваліфікаційна бакалаврська робота присвячена розробці рекомендаційної системи для ігрової платформи, яка використовує гібридний підхід на основі графових та реляційних баз даних. Основна ідея полягає у створенні рекомендаційної системи, що дозволяє формувати персоналізовані рекомендації ігор на основі їх жанрової приналежності, сеттингу, тегів та попередніх взаємодій користувача.

Система поєднує в собі можливості графової БД Neo4j (для моделювання взаємозв'язків між іграми, жанрами та користувачами) та реляційної БД PostgreSQL (для зберігання тегів і користувацьких запитів), об'єднаних за допомогою Python-бібліотеки psycopg2. Застосована клієнт-серверна архітектура дозволяє реалізувати REST API, за допомогою якого відбувається обмін даними між компонентами.

Рекомендації формуються на основі аналізу графа знань, що забезпечує гнучкий і масштабований підхід до побудови логіки пошуку подібних ігор. Особливу увагу приділено персоналізації запитів, можливості залишати рецензії та подавати пропозиції щодо нових ігор.

Новизна роботи полягає в поєднанні графового моделювання, інтеграції з реляційними джерелами даних і використанні Python для побудови динамічного алгоритму рекомендацій. Практична цінність підтверджується створенням прототипу, який може бути інтегрований у реальні платформи цифрової дистрибуції ігор або використаний як основа для розширення аналітичних функцій у сфері геймдеву.

РЕФЕРАТ

Кваліфікаційна бакалаврська робота містить 62 сторінок, 12 таблиць, 20 рисунків, список використаних джерел з 29 найменувань, 1 додатків.

«Розробка рекомендаційної системи ігрової платформи з використанням графових баз даних»

Перелік ключових слів: рекомендаційна системи, графова база даних, Neo4j, SQL, Python, FastAPI, Psycopg2, ігрова платформа, рекомендаційна система, тег, жанр, сеттінг, адаптивність, форма запиту на нову гру, front-end, користувач, компоненти.

Кваліфікаційна бакалаврська робота присвячена вирішенню актуальної задачі — створенню ефективної рекомендаційної системи для ігрової платформи, яка забезпечує персоналізований підбір ігор для користувача на основі жанрових та сеттингових уподобань. В умовах постійного зростання кількості цифрових ігор, розширення каталогів та поглиблення зв’язків між контентом, традиційні підходи до формування рекомендацій втрачають актуальність через обмеженість в обробці складних зв’язків і контекстів. Це зумовлює потребу в застосуванні нових методів, зокрема графових баз даних і гібридних моделей з’єднання різнорідних джерел даних.

Розроблена система є багаторівневим вебзастосунком, який функціонує за архітектурною моделлю “клієнт-сервер”. Клієнтська частина реалізована за допомогою бібліотеки React, що забезпечує створення адаптивного та інтерактивного інтерфейсу користувача. Серверну логіку розроблено на Python із використанням фреймворку FastAPI. Такий стек дозволяє ефективно обробляти запити, будувати REST API, а також виконувати обчислення та фільтрацію даних.

Інформаційна структура системи базується на двох типах сховищ. Для зберігання зв’язків між іграми, жанрами, сеттінгами й розробниками використовується графова база даних Neo4j. Її гнучкість у роботі зі складними структурами взаємозв’язків реалізується за допомогою мови запитів Cypher.

Додатково використовується реляційна база PostgreSQL для збереження метаданих, зокрема тегів, запитів користувачів, параметрів фільтрації та історії взаємодії. Інтеграція обох БД реалізована через бібліотеку psycopg2, що дозволяє зв’язати SQL-дані з графовим контекстом.

Ключовий функціонал системи включає авторизацію користувачів, перегляд рекомендаційних списків за жанрами і сеттінгами, виконання персоналізованих запитів, формування сторінки профілю, а також подання запиту на додавання нової гри. Для ігрових розробників та адміністраторів передбачено окремі функції керування контентом, зокрема додавання нових ігор, редагування описів і перегляд статистики.

Значну увагу приділено структурі даних, валідності інформації та оптимізації запитів. Алгоритми побудови рекомендацій реалізовані на основі графового аналізу, що дозволяє враховувати не лише прямі відповідності, але й глибші подібності між іграми через спільні жанрові та тематичні зв’язки.

Пілотне використання системи на тестовому наборі даних підтвердила працездатність прототипу та ефективність обраного підходу. Рекомендації є релевантними, а структура системи — масштабованою. Робота демонструє практичну придатність запропонованого рішення для подальшого розвитку — включно з впровадженням поведінкових моделей, збором статистики та реалізацією самонавчальних алгоритмів у майбутньому.

Об’єктом дослідження виступає процес створення рекомендаційної системи, реалізованої на поєднані графової бази даних із реляційною, що забезпечує високу швидкодію та простоту у заповнені і виведенні даних.

Предметом дослідження є методи та інструменти побудови рекомендаційних систем з використанням графових баз даних для аналізу зв'язків між іграми, жанрами, сеттінгами та тегами.

Мета бакалаврського дипломного проекту полягає в створення прототипу рекомендаційної системи, яка поєднує переваги графових і реляційних БД, з використанням модуля psycopg2 для інтеграції та запиту даних із PostgreSQL для персоналізованого підбору ігор.

Теоретична значущість роботи полягає в узагальненні та систематизації підходів до побудови гібридних інформаційних систем, які поєднують графові та реляційні моделі даних для реалізації персоналізованих рекомендацій. У роботі розглянуто особливості використання графової моделі знань на базі Neo4j та мови запитів Cypher для аналізу взаємозв’язків між іграми, жанрами, сеттінгами, що дозволяє сформувати гнучкі алгоритми підбору контенту.

Методична значущість відображається у використанні об'єктно-орієнтованого підходу до проєктування підсистем, впровадженні REST-архітектури за допомогою FastAPI, а також у застосуванні бібліотеки psycopg2 для побудови механізмів обміну даними між PostgreSQL та графовою БД Neo4j. Робота демонструє практичне застосування концепцій RAD-розробки, включаючи швидке прототипування, модульність та повторне використання коду.

Практична значущість проєкту підтверджується створенням повноцінної рекомендаційної системи, яка може бути використана як прототип для комерційних платформ цифрової дистрибуції ігор. Реалізовані механізми аналізу жанрово-сеттингових зв’язків, обробки запитів та виводу персоналізованих рекомендацій є прикладом інтеграції сучасних підходів до роботи з великими обсягами взаємопов’язаної інформації у сфері розважального програмного забезпечення.

Рік виконання кваліфікаційної бакалаврської роботи – 2025.

Рік захисту роботи – 2025.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі ринок відеоігор демонструє динамічне зростання, що супроводжується щорічним збільшенням кількості нових ігор, особливо інді-проєктів. Це створює надлишок контенту, що ускладнює орієнтацію для користувачів. У такому контексті особливого значення набувають рекомендаційні системи, які забезпечують персоналізований підбір контенту відповідно до інтересів користувача. Актуальність теми обумовлена потребою у впровадженні гнучких рішень, що поєднують сучасні бази даних і механізми пояснюваних рекомендацій.

Предметна галузь проєкту охоплює інформаційні технології в контексті цифрової дистрибуції ігор. Об’єктом дослідження виступає процес створення гібридної рекомендаційної системи, яка використовує графову базу Neo4j для зберігання зв’язків між іграми, жанрами, сеттінгами та реляційну базу PostgreSQL для зберігання тегів, що дозволяє підвищити точність рекомендацій.

Особливістю розробленої системи є поєднання графових і реляційних підходів, що дозволяє зберігати гнучку структуру і надавати обґрунтування рекомендацій. Реалізовано API на базі FastAPI, а також користувацький інтерфейс із можливістю фільтрації ігор за жанром, тегами та сеттінгом, побудований за допомогою React.

Метою дипломного проєкту є проєктування, моделювання та реалізація рекомендаційної системи, яка забезпечує персоналізований підбір ігор на основі структурованих зв’язків і метаданих, з використанням сучасних технологій зберігання, обробки та виведення інформації.

Відповідно до поставленої мети визначено такі завдання:

- проаналізувати предметну галузь, досвід застосування існуючих рекомендаційних систем;
- визначити функціональні та нефункціональні вимоги до майбутньої системи;

- побудувати архітектуру проєкту з урахуванням потреб персоналізації;
- створити графову модель даних у Neo4j та реляційну модель тегів у PostgreSQL;
- реалізувати API для обробки запитів і формування рекомендацій;
- реалізувати інтерфейс користувача для взаємодії із системою;
- протестувати прототип системи і сформувати висновки щодо його ефективності;
- оцінити потенціал масштабування і застосування розробленого рішення.

У процесі виконання роботи використовувались такі методи дослідження: структурно-логічний і порівняльний аналіз наявних систем, об'єктно-орієнтоване моделювання (UML, сценарії), а також вербально-описовий метод для представлення архітектурних рішень.

Практична значущість роботи полягає в тому, що реалізовано діючий прототип рекомендаційної системи, який може бути інтегрований у цифрову платформу для пошуку ігор, а також слугувати основою для стартапів або освітніх проєктів.

Теоретична значущість полягає в дослідженні поєднання графових і реляційних моделей у межах єдиної системи, що відкриває перспективи для впровадження нових архітектур у сфері персоналізованих рекомендацій.

Робота складається з трьох розділів. Перший містить аналіз предметної галузі та літературних джерел. порівняння сучасних рекомендаційних підходів. У другому розділі постановку задачі, описано архітектуру системи, побудовано моделі даних.. Третій розділ присвячений реалізації системи, її тестуванню та оцінці результатів.

РОЗДІЛ 1

ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі та об’єкта дослідження

Предметна галузь проєкту охоплює сферу розробки інтелектуальних інформаційних систем у контексті цифрової дистрибуції відеоігор. Це одна з найактуальніших і швидкозростаючих галузей у сучасному IT-середовищі, що охоплює процеси аналізу, структурування та рекомендації контенту в умовах надлишку інформації. У контексті цього дослідження вона охоплює підходи, технології та інструменти, які застосовуються для створення рекомендаційних систем на основі взаємозв’язків між іграми, жанрами, сеттінгами та тегами.

Об’єктом дослідження є процес розробки рекомендаційної системи для ігрової платформи, яка базується на використанні графової бази даних Neo4j для збереження ієрархічних і семантичних зв’язків, та реляційної бази PostgreSQL для керування тегами та атрибутами ігор. Застосовується гібридний підхід, що дозволяє інтегрувати обидві моделі через Python-модуль Psycopg2.

Ігрова рекомендаційна система — це програмна платформа, що надає користувачеві персоналізовані пропозиції ігор, базуючись на його вподобаннях, жанрових і тематичних зв’язках між іграми, а також тегах, сформованих за жанром і сеттінгом. У контексті конкуренції на ринку цифрової дистрибуції, такі системи сприяють покращенню користувацького досвіду, утриманню аудиторії та збільшенню ефективності пошуку релевантного контенту.

Серед об’єктів предметної галузі можна виокремити:

- Користувачів, які взаємодіють з ігровими платформами;
- Розробників ігор, зацікавлених у просуванні контенту;
- Платформи цифрової дистрибуції (Steam, Epic Games, GOG тощо);
- Бази даних (Neo4j, PostgreSQL), які забезпечують зберігання контенту та метаінформації;

- Інструменти бекенд- та фронтенд-розробки (FastAPI, React, py2neo, psycopg2 тощо).

Серед об’єктів предметної галузі, що охоплює сферу розробки рекомендаційних систем для ігрових платформ, варто зазначити кілька ключових компонентів. По-перше, це користувачі, які безпосередньо взаємодіють із платформами цифрової дистрибуції, обираючи та оцінюючи ігровий контент[1]. По-друге, до важливих суб’єктів належать розробники ігор, зацікавлені у просуванні свого продукту серед цільової аудиторії. Значну роль відіграють також самі платформи дистрибуції, такі як Steam, Epic Games чи GOG, які виступають посередниками між контентом і споживачем.[3]

Інформаційна інфраструктура предметної галузі представлена базами даних, зокрема графовими (Neo4j) і реляційними (PostgreSQL), які забезпечують зберігання і структуризацію як основного контенту, так і супутньої метаінформації. Інструменти для бекенд- і фронтенд-розробки (FastAPI, React, а також бібліотеки py2neo і psycopg2) дають змогу реалізувати повний цикл створення веборієнтованої рекомендаційної системи.

У цій галузі приймаються типові технічні рішення, що охоплюють проектування структури графової моделі, яка відображає взаємозв’язки між іграми, жанрами, сеттінгами, а також побудову реляційних залежностей між тегами та характеристиками контенту. Важливим етапом є розробка алгоритмів формування рекомендацій з урахуванням користувацьких вподобань. Значна увага приділяється побудові API для забезпечення ефективної взаємодії з клієнтськими застосунками, а також створенню інтерфейсів, які б забезпечували зрозуміле подання рекомендацій користувачеві.

На прийняття рішень у цій сфері впливають різноманітні чинники. Серед них — стрімке зростання обсягу ігрового контенту, що ускладнює навігацію користувача; зростаюча потреба у персоналізованому пошуку; поширення відкритих технологій зберігання складних зв’язків; підвищені вимоги до пояснюваності результатів рекомендацій та загальна потреба у прозорих і масштабованих системах. Вибір бази даних визначає архітектуру системи, а тип

алгоритму — логіку рекомендацій. Залежно від архітектури API формується механізм передачі результатів до кінцевого користувача. Таким чином, кожне прийняте рішення визначає масштабованість, ефективність і зручність взаємодії з системою.

Для прийняття обґрунтованих рішень використовується як емпірична інформація (аналітика поведінки користувачів, статистика переглядів, показники залученості), так і дані з перевірених джерел, таких як офіційна документація до Neo4j і PostgreSQL, наукові публікації з тематики графового моделювання та рекомендаційних систем, а також практичні кейси побудови гібридних моделей.

Методологія розробки системи ґрунтується на використанні графового підходу до представлення даних, написанні запитів мовою Cypher, алгоритмізації процесів рекомендації та аналізу мережевих зв’язків. Компонентна архітектура дозволяє чітко відокремити підсистеми API, інтерфейсу та баз даних, що забезпечує гнучкість у масштабуванні та подальшому розвитку системи.

Тип задач, які розв’язуються в межах предметної галузі, є відкритими та комплексними, такими, що потребують адаптації до постійно змінюваних вхідних даних. Система повинна підтримувати динамічну інтеграцію між джерелами інформації, масштабованість і здатність до модифікації в міру зростання кількості контенту або користувачів. У межах даного дослідження ці задачі вирішуються шляхом прототипування архітектури, моделювання графа ігрового контенту та поєднання графових і реляційних структур даних.

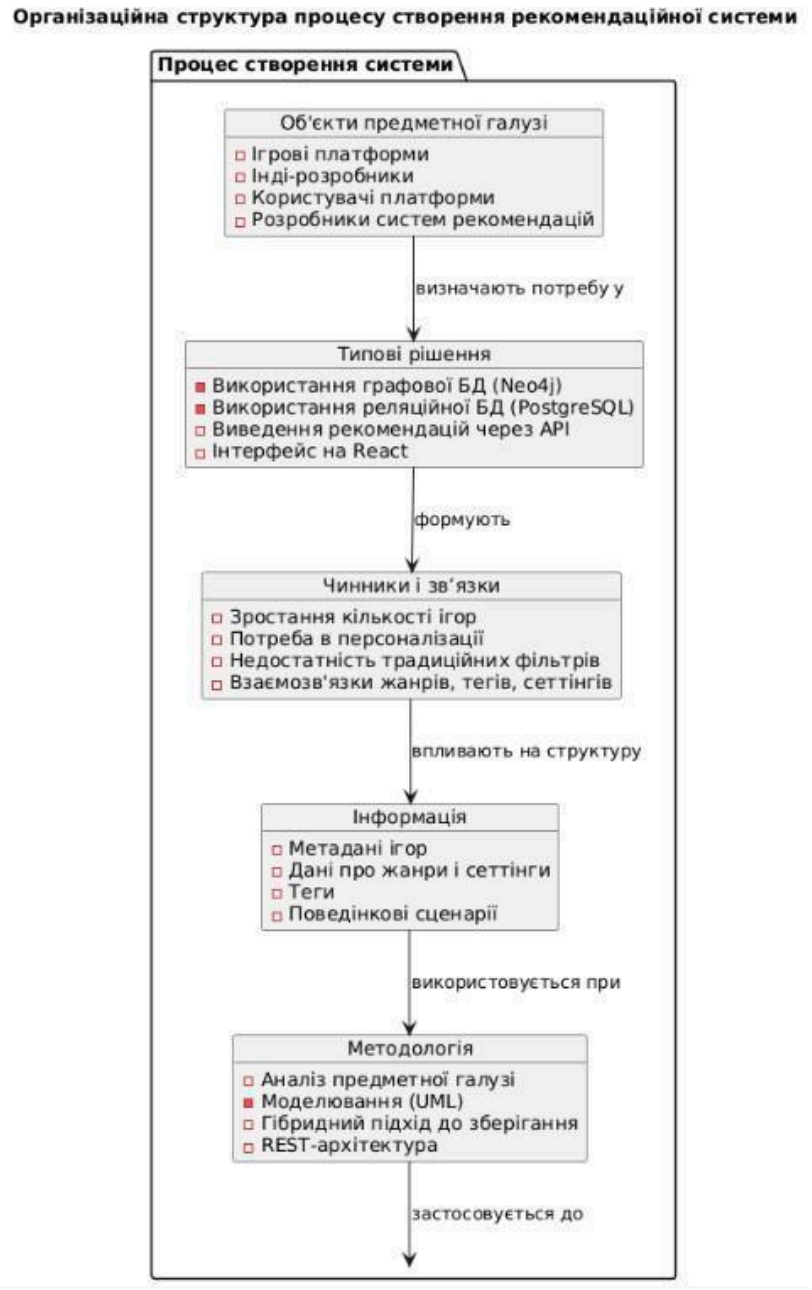

Рисунок 1.1.1 - Організаційна структура процесу створення рекомендаційної системи

1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Сфера побудови інтелектуальних рекомендаційних систем у цифровому середовищі

— одна з ключових тем сучасної прикладної інформатики та програмної інженерії. Особливої актуальності ця галузь набуває у сфері цифрової дистрибуції ігор, де

користувачі стикаються з інформаційним перевантаженням, а розробники потребують ефективних засобів просування контенту. Побудова рекомендацій, які враховують зв’язки між жанрами, сеттінгами, тегами й іншими властивостями ігор, потребує залучення новітніх технологій зберігання та обробки даних, зокрема графових баз даних.

Аналіз сучасних літературних джерел свідчить про стійку тенденцію до використання графових структур у системах рекомендацій. У роботі Wang et al. (2021)[1] «Graph Learning based Recommender Systems: A Review» наголошується на ефективності графових моделей у складних предметних доменах, де необхідно враховувати багаторівневі зв’язки між сутностями. Дослідження Schlichtkrull et al. (2018) також демонструє успішне застосування графових нейронних мереж (GNN) для рекомендацій на основі знань (knowledge graphs).[2]

Такі системи як Netflix, Amazon, Spotify частково вже впроваджують графову логіку для персоналізації контенту. Проте, у відкритому доступі недостатньо прикладних реалізацій для геймінг-сфери, що створює перспективу для академічних і проєктних рішень.

Аналіз літературних джерел та практичного досвіду у сфері побудови рекомендаційних систем показує наявність кількох основних підходів до реалізації таких рішень у цифровому середовищі:

- Колаборативна фільтрація — один із найпоширеніших методів, що базується на поведінковій подібності користувачів (покупки, оцінки, час перегляду). Підходить для великих платформ, проте має недолік "холодного старту" (cold start problem) та обмежену пояснюваність.
- Контентна фільтрація — формує рекомендації на основі властивостей контенту, таких як жанр, платформа, опис тощо. Простий у реалізації, але не враховує соціальний контекст чи популярність серед схожих користувачів.
- Гібридні моделі — поєднують обидва попередні методи, дозволяючи компенсувати їх недоліки, однак такі рішення часто є

складнішими у впровадженні та вимагають гнучких структур даних.

- Графові рекомендаційні системи — відносно новий напрям, що набуває популярності завдяки можливості представлення складних зв’язків у вигляді вузлів і ребер. Наприклад, гра може мати зв’язки з жанрами, тегами, розробниками, сеттінгами, а запити до графа дозволяють виявити приховані зв’язки та схожість.

Недоліками традиційних реляційних моделей є їхня обмежена здатність ефективно відображати багатозв’язкові структури. Графові бази, такі як Neo4j, навпаки, дозволяють легко реалізувати запити на кшталт: "знайти всі ігри, пов’язані з жанром RPG і сеттінгом Fantasy, що мають спільні теги з грою X". Це суттєво підвищує релевантність і пояснюваність рекомендацій.[6]

Серед основних інформаційних потоків у системі можна виокремити:

- Вхідні дані — запити користувача за параметрами (жанр, сеттінг, дата випуску), а також дії в інтерфейсі;
- Вихідні дані — список рекомендованих ігор із тегами, що пояснюють, чому їх запропоновано;
- Системні запити — звернення до Neo4j для отримання структури зв’язків та до PostgreSQL для витягування тегів, що базуються на жанрах і сеттінгах.

Важливим є чіткий розподіл обов’язків між компонентами системи:

- Neo4j обробляє запити, що стосуються логіки рекомендацій та схожості;
- PostgreSQL зберігає тегову інформацію та допоміжні таблиці;
- FastAPI відповідає за маршрутизацію та логіку об’єднання даних;
- React забезпечує зручний, адаптивний інтерфейс взаємодії з користувачем.

Основна цільова аудиторія таких систем:

- Гравці, які прагнуть швидко знаходити ігри відповідно до своїх інтересів;
- Платформи цифрової дистрибуції, що потребують ефективного механізму персоналізації;

- Розробники, які зацікавлені в більш релевантному просуванні своїх проєктів.

Практичний досвід показує, що подібні системи ефективно працюють при поєднанні гнучкого фронтенду (React, Tailwind CSS) з потужним бекендом на основі FastAPI, Neo4j і PostgreSQL. Завдяки цьому забезпечується високий рівень масштабованості, адаптивності та пояснюваності рекомендацій.

РОЗДІЛ 2

РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Аналіз і специфікація вимог до інформаційної системи/підсистеми

Процес визначення вимог є критично важливим етапом у розробці будь-якої інформаційної системи, що забезпечує її цілеспрямовану, ефективну та передбачувану реалізацію. Вимоги — це умови, які повинна задовольняти система або її компоненти для досягнення поставленої мети, забезпечення працездатності, відповідності технічним стандартам і задоволення очікувань кінцевого користувача.

Ієрархія бізнес-вимог (діаграма вимог типу «requirement diagram»)

Головна бізнес-вимога: BR1. Надати користувачам доступ до інформації про гру

- BR1.1: Відображати головну інформацію про гру (Назва, теги, жанри та сеттінги, посилання на платформи)
- BR1.2: Надати розділ «Профіль» (список переглянутих проєктів, відслідковувані проєкти, нові рекоменадації та рецензії, тощо)
- BR1.3: Проінформувати про актуальність проєктів
- BR1.4: Інтегрувати форму запитів нових ігор на платформу Зв’язки з іншими елементами моделі
    - Джерело вимог: потенційний користувач сайту, розробник
    - Бізнес-цілі: особистий маркетинг, просування ігор
    - Обмеження: використання лише відкритих технологій (Python,Neo4j, SQL), некомерційне застосування

Таблиця 1.1 – Бізнес-цілі

Стейкхолдер	**Бізнес-ціль**	**Бізнес-вимоги до ІС**
Замовник (платформа)	Презентація ігор	Сайт має містити окремі розділи: "Пошук ігор", "Оновлення", "Рекомендації", "Профіль"
Користувачі	Легко знайти інформацію	Сайт має мати навігацію з чітким меню та адаптивний дизайн для мобільних пристроїв
Розробники	Легкість просування гри	На головній сторінці мають бути стисло представлені ключова інформація про гру та посилання на платформи де можливо отримати доступ
Технічний консультант	Перевірка інформаці	Інформація про гру, теги, описи та посилання повинні бути актуальними і бути на офіційних платформах цифрової дистрибуції

Користувацькі вимоги

Аналіз потреб користувача показує, що важливо забезпечити інтуїтивно зрозуміла та логічна навігація між розділами інтерфейсу, швидке завантаження запитів, коректне відображення на різних пристроях. Дизайн має бути естетичним і адаптивним. Також необхідна функціональна форма запитів з перевіркою на помилки та підтвердженням успішного надсилання. Робота системи не повинна вимагати встановлення додаткового програмного забезпечення.

Таблиця 1.2 – Функціональні вимоги

Стейкхолдер	**Бізнес-ціль**	**Бізнес-вимоги до ІС**
FR1	Головна сторінка	Відображення основної інформації про користувача
FR2	Розділ «Профіль»	Надання інформації про переглянуті та відслідковані проєкти та час використання платформою
FR3	Розділ «Дослідження»	Перелік нових ігор, які нещодавно розміщенні або подібні до попередніх запитів
FR4	Форма запиту	Надсилання запитів на новий проєкт через форму
FR5	Адаптивність	Коректне відображення на мобільних пристроях

Таблиця 1.3 – Нефункціональні вимоги

Ідентифікатор	**Назва**	**Повна назва**	**Статус**	**Складність**	**Пріоритет**	**Ризик**	**Примітки**
NFR1	Продуктивність	Запит повинен завантажуватись за ≤ 10 сек	Затверджено	Висока	Високий	Високий	Оптимізація коду
NFR2	Надійність	Система не повинна давати збої при переході між запитами	Затверджено	Середня	Високий	Середній	Тестування навігації
NFR3	Безпека	Забезпечення базового захисту від спаму у формі запитів	Заплановано	Середня	Середній	Високий	CAPTCHA
NFR4	Універсальність	Підтримка популярних платформ (Steam,GOG, Epic Games,)	Затверджено	Низька	Середній	Низький	Кросбраузерність

NFR5	Масштабованість	Легке додавання нових розділів і компонентів	Затверджено	Висока	Високий	Середній	Компонентна архітектура React

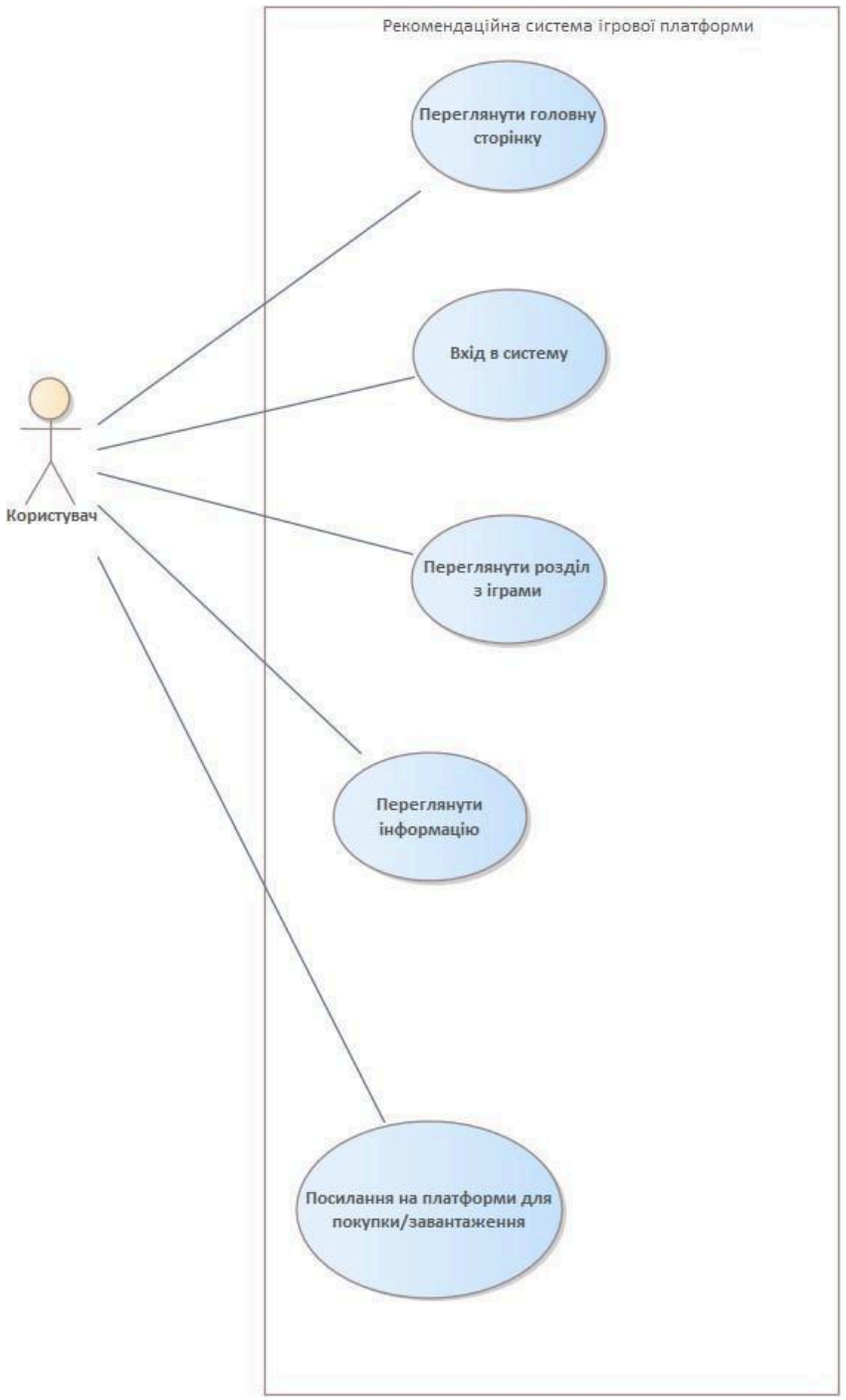

Рисунок 1.1 - Діаграма вимог UseCase діаграма

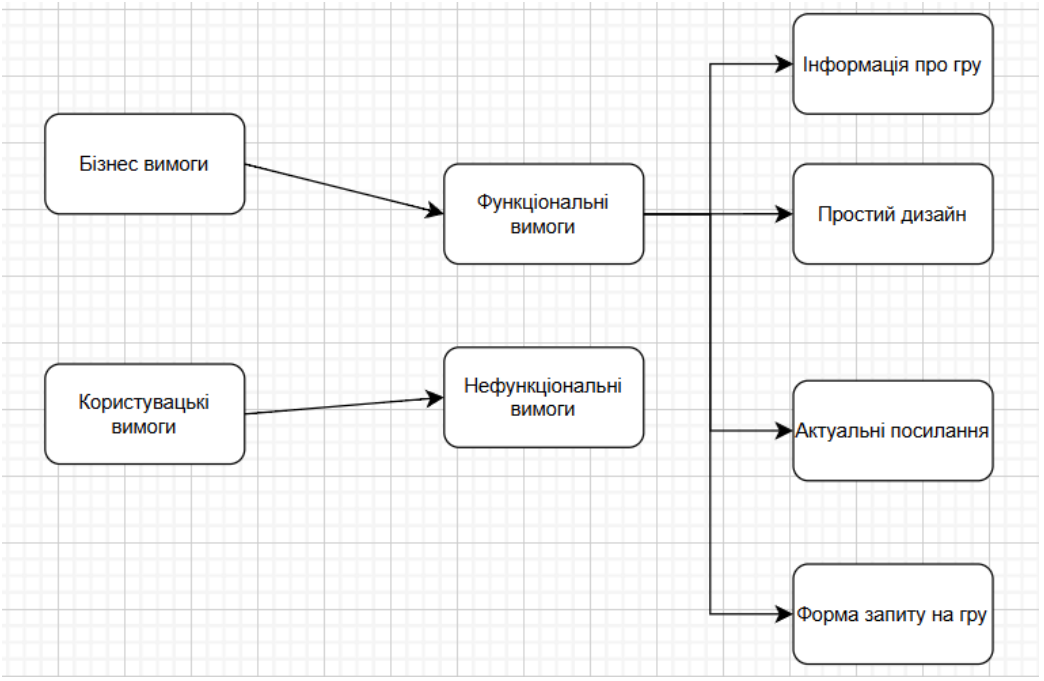

Рисунок 1.2 - Діаграма зв’язків між вимогами та елементами системи

Характеристики вимог

Усі вимоги сформульовані згідно з принципами якості: вони є чіткими, завершеними, послідовними, не суперечать одна одній, мають однозначне тлумачення та можуть бути перевірені після реалізації. Наприклад, вимога щодо темного режиму має чіткий критерій перевірки — наявність перемикача і візуальна зміна кольорової схеми.

У процесі розробки використовуються такі методи:

- структурно-логічний аналіз для формування архітектури;
- вербально-описовий підхід для формулювання вимог;
- моделювання (UML/SysML) — для представлення системних компонентів і зв'язків між ними.

Розроблена специфікація вимог стане основою подальшого проєктування інформаційної підсистеми та реалізації її компонентів у рамках дипломного проєкту.

2.2 Постановка та алгоритм розв’язання задачі

2.2.1 Постановка задачі

Метою задачі є розробка інформаційної підсистеми — рекомендаційної системи ігрової платформи — яка дозволяє користувачам отримувати персоналізовані рекомендації ігор на основі жанрів, сеттінгів і тегів, а також взаємодіяти з платформою через вебінтерфейс. Розв’язання задачі передбачає автоматизацію обробки запитів користувача, використання графової моделі даних для встановлення зв’язків між іграми та створення відповідної архітектури для генерації релевантних результатів.

Техніко-економічна сутність полягає у створенні ефективного та масштабованого цифрового інструменту для навігації у великому обсязі ігрового контенту без необхідності втручання адміністратора або зовнішнього фахівця. Реалізація задачі засобами інформаційної системи дозволяє скоротити час на пошук ігор, підвищити точність рекомендацій, забезпечити прозорість механізму та комфортність користування. Об’єкти, за управління якими розв’язується задача:

- графова база даних Neo4j (вузли: гра, жанр, сеттінг, зв’язки між ними);
- реляційна база PostgreSQL (таблиці тегів і їх зв’язків);
- форма пошуку та запиту на гру;
- API для побудови запитів і обробки відповіді;
- механізм інтеграції з базами даних (через psycopg2).

Використання вихідної інформації: використовується для побудови графа ігор, зв’язків між жанрами, сеттінгами й тегами, а також для представлення рекомендацій користувачу у зрозумілому вигляді.[5]

Періодичність: задача виконується щоразу при зверненні користувача до системи — при формуванні запиту та отриманні рекомендацій.

Умови припинення: розв’язання задачі завершується після формування результату або припинення сесії користувача.

Зв’язки з іншими задачами: задача пов’язана з формуванням запитів до баз даних, їх обробкою на сервері, збереженням зв’язків у графовій структурі, аналізом тегів та виведенням результатів на клієнт.

Розподіл дій: персонал відповідає за наповнення бази даних, тестування системи та адміністрування інтерфейсу. Технічні засоби забезпечують генерацію запитів, обробку результатів, представлення інтерфейсу та роботу API.

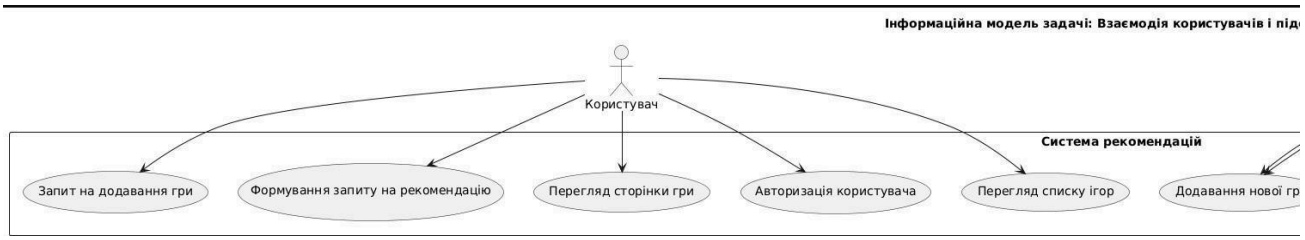

Рисунок 2.1.1 –Інформаційна модель задачі

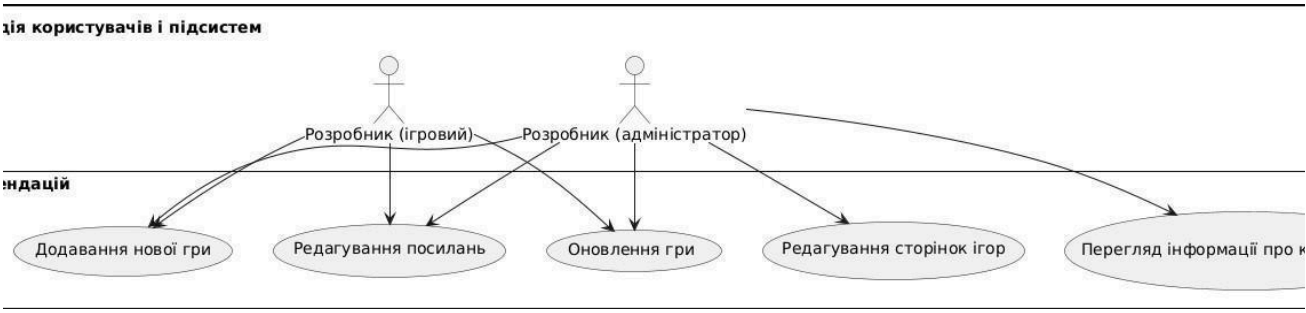

Рисунок 2.1.2 - Наступний фрагмент інформаційної моделі задачі

Вихідна інформація — це дані, які виводяться на екран користувачеві під час взаємодії з системою: вміст сторінки гри, повідомлення про успішне або помилкове надсилання запиту, статус теми (світла/темна), навігація.

Таблиця 2.1 – Вихідна інформація

№ з/п	Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувачі інформації
1	Повідомлення про результат запиту	OUT1	Виводиться в модальному вікні	При надсиланні форми	Після обробки запиту	Користувач
2	Повідомлення про помилку	OUT2	Виводиться на сторінці	У разі помилки	Після обробки запиту	Користувач
3	Список ігор	OUT3	Список карток/блокі в	Постійно	Після завантаження сторінки	Усі
4	Повідомлення про від правку запиту на нову гру	OUT4	Виводиться на сторінці	На запит	Миттєво	Користувач

Вхідна інформація — це дані, які користувач вводить або які отримуються системою автоматично: ім’я, email, повідомлення з форми, вибір теми, дані навігації.

Таблиця 2.2 – Вхідна інформація

№ з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	Дані форми	IN1	Текстові поля	Кожне заповнення	Користувач
2	Дії навігації	IN2	Події кліків	Будь-який момент	Користувач
3	Запит пошуку	IN3	JSON	На запит користувача	користувач
4	Дані авторизації (опц.)	IN4	Token/URL (у API стороніх платформ)	У разі інтеграції	API стороніх платформ

2.2.2 Алгоритм розв’язання задачі

У межах реалізації рекомендаційної підсистеми ігрової платформи використовуються різні типи вхідної інформації. Частина цих даних надходить безпосередньо від користувача (дані з форми запиту нової гри, авторизаційні дані, параметри навігації, взаємодія з інтерфейсом), інша — отримується через запити до зовнішніх або внутрішніх баз даних (інформація про ігри, жанри, сеттінги, рецензії, активність користувача тощо). Уся інформація передається, обробляється та зберігається у форматах, придатних для обробки за допомогою графових (Cypher-запити) і реляційних (SQL-запити) підходів.

Таблиця 2.3 – Перелік масивів використаної інформації

№ з/п	Масив (назва колекції MongoDB)	Ідентифікатор	Максимальна кількість записів
1	Дані з форми запиту	IN_RequestForm	100
2	Відомості про ігри	IN_Games	10 000
3	Дані користувача (профіль)	IN_User	1
4	Жанри та сеттінги	IN_Meta	200
5	Навігаційні події	IN_Nav	50

Результати обчислень у системі формуються у вигляді персоналізованих добірок ігор, повідомлень про успішне виконання дій (авторизація, відправка форми), списків рекомендованих або популярних ігор, та виводу інформації про конкретну гру. Результати також включають графові візуалізації зв’язків між іграми, жанрами та тегами. Всі дані відображаються безпосередньо у вебінтерфейсі та адаптуються до користувацьких уподобань.

Таблиця 2.4 – Перелік масивів результатної інформації

№ з/п	Масив	Ідентифікатор	Максимальна кількість записів
1	Повідомлення про успішну дію	OUT_Success	1
2	Повідомлення помилки	OUT_Error	10
3	Відображені ігри у результатах	OUT_Games	50
4	Добірки за жанром/сеттінгом	OUT_Recommend	20
5	Візуалізація графових зв’язків	OUT_Graph	1000 вершин/ребер

Математичний опис: Задача розробки рекомендаційної системи базується переважно на алгоритмічних і логічних принципах побудови зв’язків у графі. Для окремих підсистем (наприклад, валідації форми запиту чи фільтрації добірок) можуть використовуватись елементи математичної логіки, фільтрації та класифікації.

Приклад логічної формалізації вибору ігор за тегами:

$$R(u) = \{g \in G \mid \exists t \in T_u : (g)-[:HAS\_TAG]->(t)\}$$

де:

- u — користувач;
- T_u — множина тегів, з якими взаємодіяв користувач;
- g — гра з множини всіх ігор G;
- R(u) — множина рекомендованих ігор;
- HAS_TAG — тип ребра у графі, що пов’язує гру з тегом.

Функція валідації форми запиту на нову гру:

$$Valid = (Name \neq \varnothing) \land (Email \in \mathscr{E}) \land (GameTitle \neq \varnothing) \land (Link \rightarrow URL) \land (Message.length \leq 500)$$

де $\mathscr{E}$ — множина валідних email-адрес.

2.3 Моделювання рекомендаційної системи

Моделювання інформаційної системи є важливим етапом розробки рекомендаційної системи, що дозволяє формалізувати поведінку та структуру системи до початку її реалізації. Основу моделювання становить мова UML, яка забезпечує єдину візуальну мову опису різних аспектів системи, включаючи взаємодію користувача з інтерфейсом, логіку функціонування та компонування системних елементів.

Метод моделе-орієнтованого інжинірингу (Model-Driven Engineering, MDE) дає змогу описати архітектуру системи через набір узгоджених моделей, які далі використовуються для проектування, аналізу та валідації архітектурних рішень.

2.3.1 Моделювання поведінки системи

Діаграма прецедентів

На діаграмі прецедентів відображено взаємодію основних акторів (користувач, розробник, адміністратор) із системою. Основні варіанти використання: перегляд інформації, вхід у систему, перегляд ігор, посилань на платформи та редагування інформації на сторінках ігор.

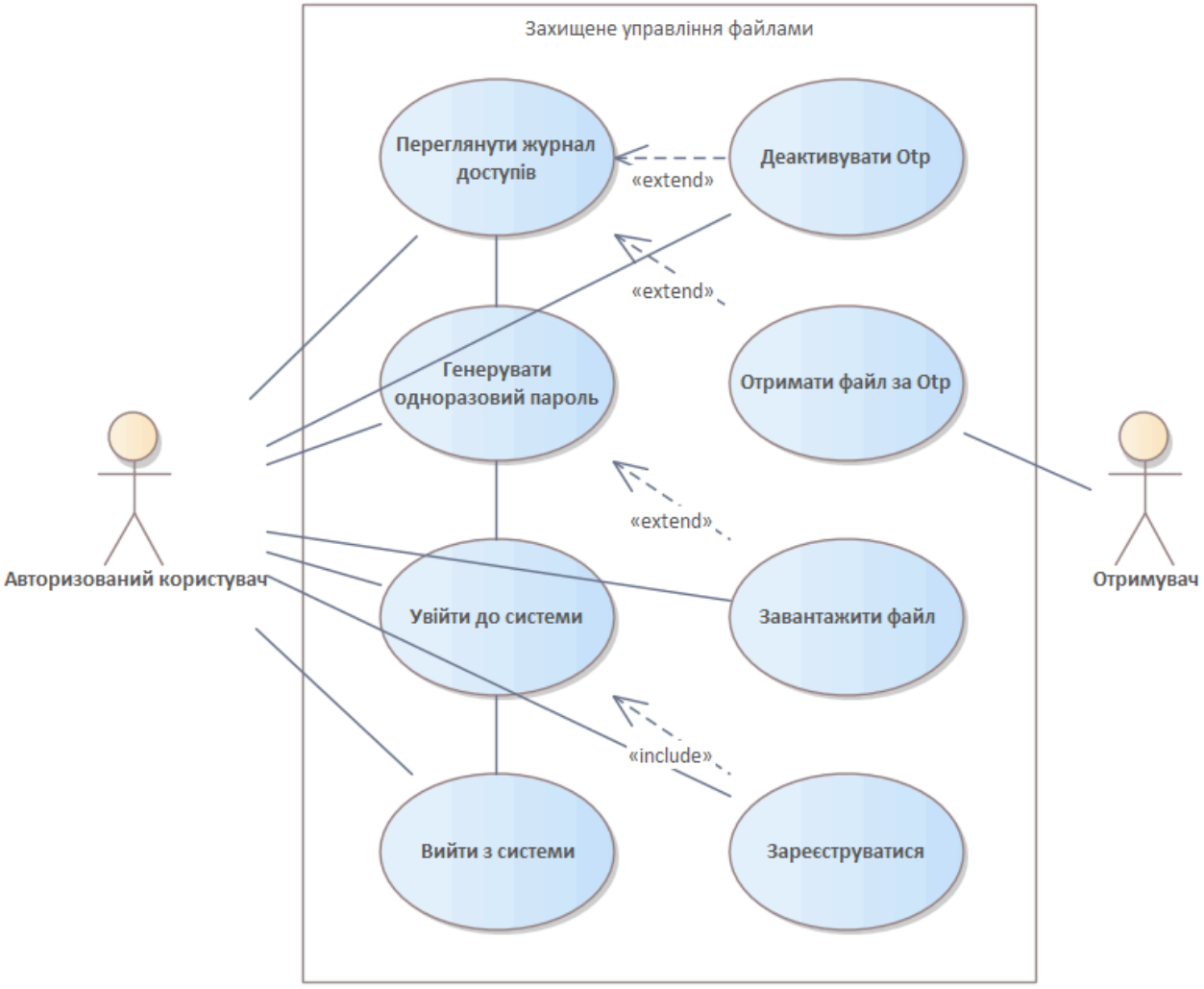

Рисунок 3.1 – Діаграма прецедентів рекомендаційної системи

Діаграми послідовності

Для кожного з визначених прецедентів створено діаграми послідовності, які демонструють поетапну взаємодію об’єктів у межах

системи. У випадку надсилання форми зворотного зв’язку послідовність виглядає так: користувач взаємодіє з формою, далі відбувається валідація введених даних, після чого запит надсилається на сервер і обробляється відповідь.

У процесі перегляду проєктів маршрутизатор спрямовує запит до відповідного компонента, який виконує виведення результатів запит

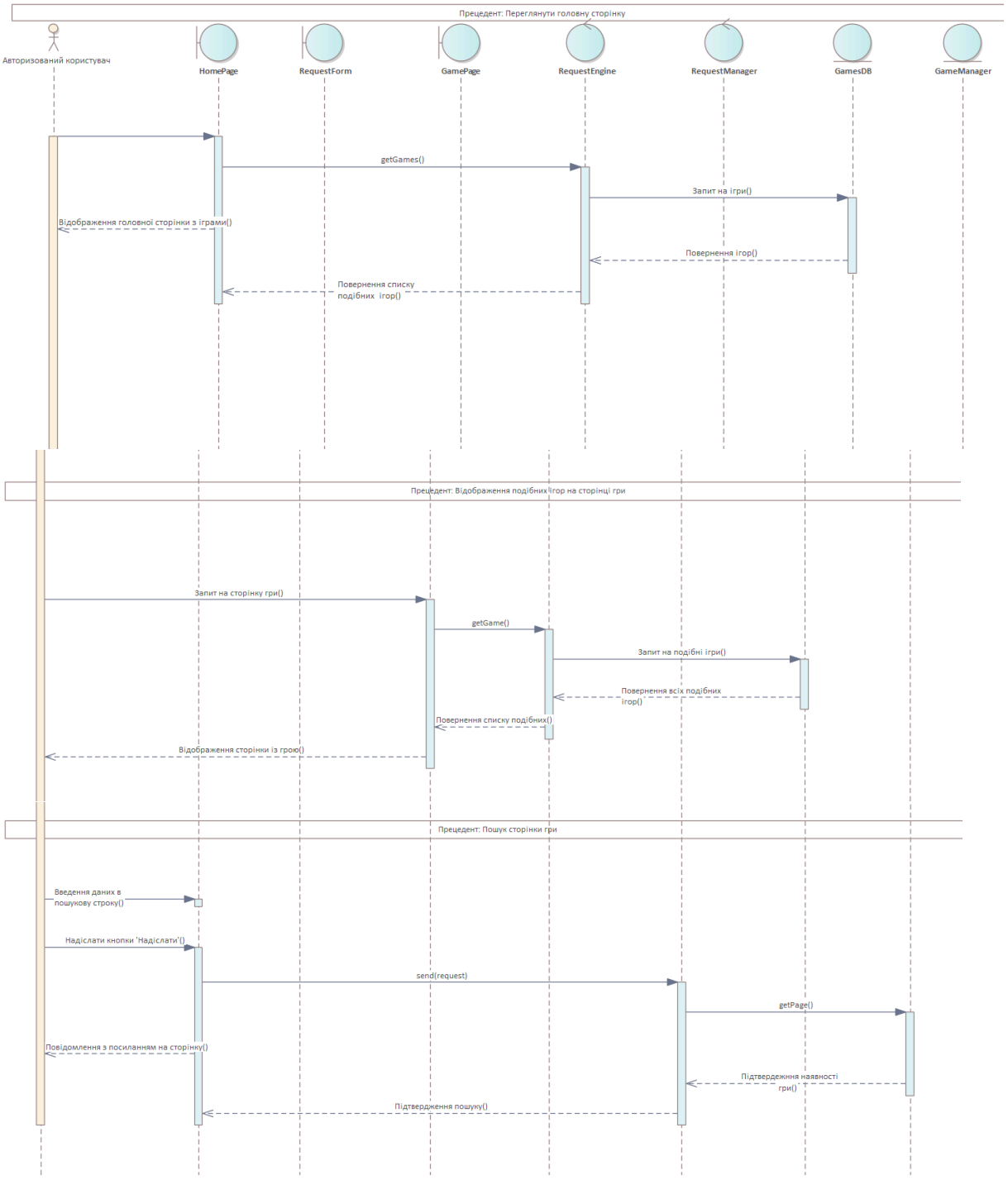

Рисунок 3.2 – Діаграма послідовності

Діаграма діяльності

У рамках загального процесу взаємодії з системою (від завантаження до взаємодії) створено діаграму діяльності, що демонструє всі гілки роботи системи.

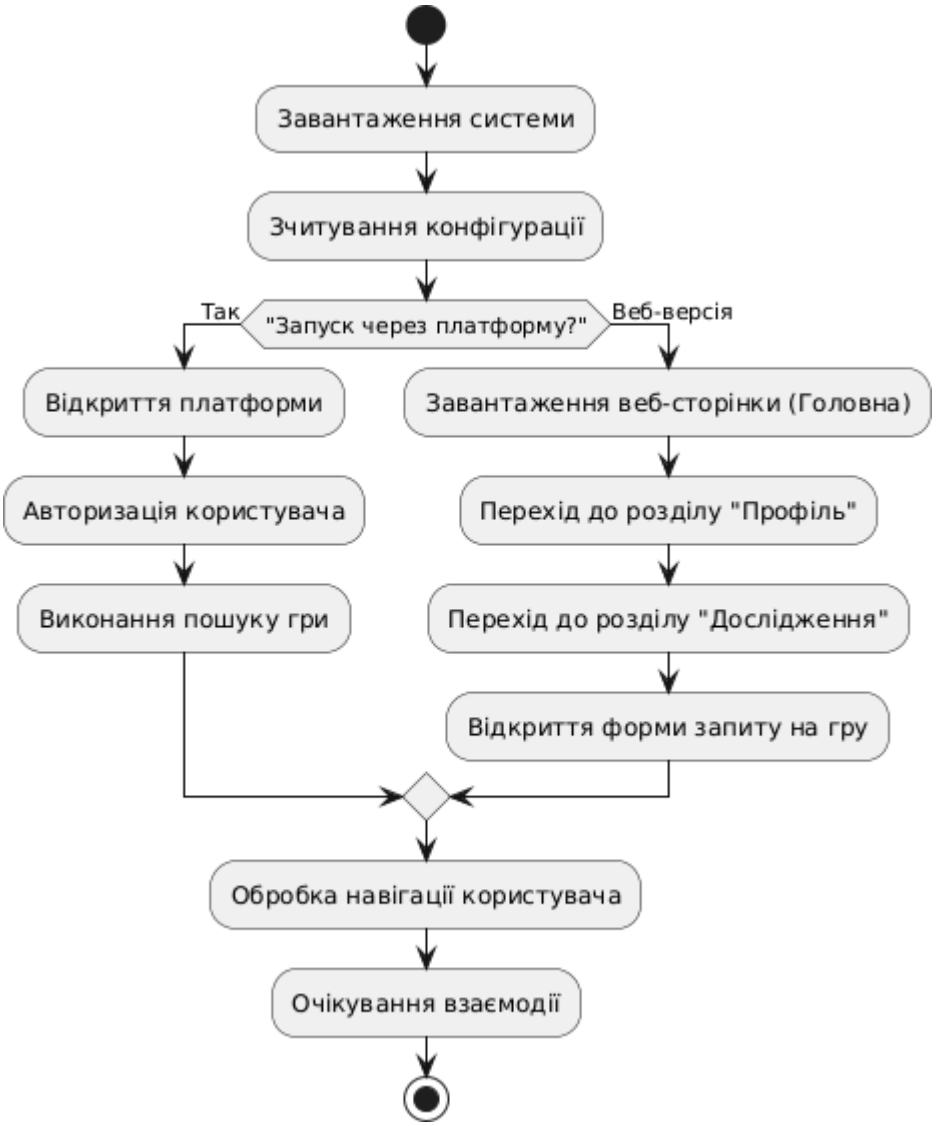

Рисунок 3.3 – Діаграма діяльності

2.3.2 Моделювання структури застосунку

Діаграма класів

Модульна структура сайту реалізована за допомогою компонентів React. На діаграмі класів відображено:

- Класи-сутності: Game, Request, Genres and Settings, Tags
- Граничні класи (інтерфейс): RequestForm, ProffilePage, NavBar,

GamePage, Footer, HomePage

- Класи-менеджери: RequestManager, UIController, GameManager

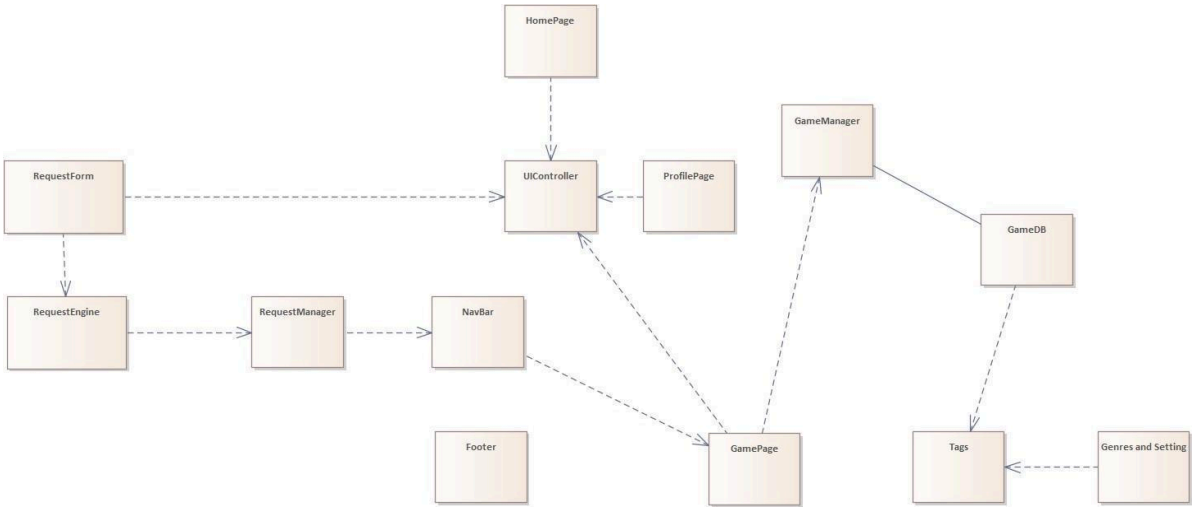

Рисунок 2.3.4 – Діаграма класів

РОЗДІЛ 3

ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ЗАСТОСУНКУ

3.1 Інформаційне забезпечення

Інформаційне забезпечення (ІЗ) системи охоплює структурування, зберігання, обробку та передачу даних у межах рекомендаційної системи ігрової платформи. Воно побудоване з урахуванням вимог до швидкодії, масштабованості та логічної цілісності даних, які характеризують взаємозв’язки між іграми, жанрами, сеттінгами, тегами та користувачами.

Інформаційне забезпечення рекомендаційної системи реалізовано у вигляді гібридного підходу: використовується графова база Neo4j, яка містить вузли (ігри, жанри, сеттінги, користувачі) та ребра зв’язків (HAS_GENRE, HAS_SETTING, PLAYED_BY, RECOMMENDED), а також реляційна база PostgreSQL, у якій зберігаються тегові структури, що дозволяють деталізувати опис гри на основі жанрово-сеттінгового контексту.

Доступ до реляційної бази даних здійснюється через модуль psycopg2 на стороні Python-сервера, що забезпечує об’єднання даних з обох джерел і формування результатів на запити користувачів.

Для зберігання та обробки даних використовуються такі програмні засоби:

- Cypher-запити — для побудови, оновлення та читання даних у Neo4j;
- SQL-запити (PostgreSQL) — для доступу до таблиць tags, genre_tag, setting_tag;
- Python — як серверна логіка обробки запитів, що зв’язує обидві бази даних;
- JSON — формат даних для обміну між frontend і backend;
- HTTP API — для надсилання користувацьких запитів (наприклад, пошуку ігор за жанром/сеттінгом або створення рецензії).

У системі забезпечено логічне розділення інформації:

- графова модель містить дані про самі ігри та зв’язки між ними;
- реляційна база — класифікацію тегів, які пов’язуються із жанрами та сеттінгами, що дає змогу побудувати систему персоналізованих описів і пояснень до рекомендацій.

Основні вхідні дані формуються під час взаємодії користувача із системою: запити на пошук ігор, авторизація, оцінки, коментарі, налаштування профілю. Ці дані надходять у backend через REST API, обробляються Python-модулем і, в залежності від запиту, спрямовуються або до Neo4j, або до PostgreSQL.

Вихідні дані — це:

- сформовані списки ігор за жанром, тегами або сеттінгами;
- персоналізовані рекомендації;
- детальна інформація про гру (опис, дата релізу, жанри, сеттінг, теги);
- повідомлення про успішне або помилкове виконання дії (наприклад, надсилання запиту на нову гру).

3.2 Технічне забезпечення

Технічне забезпечення є сукупністю програмно-апаратних засобів, призначених для функціонування інформаційної системи. Цей розділ описує комплекс технічних засобів (КТЗ), що забезпечує збір, обробку, зберігання та передачу інформації в системі захищеного завантаження файлів, а також обґрунтовує їх вибір та розташування.

3.2.1 Загальні положення та схема автоматизації

Пропонована інформаційна система для побудови персоналізованих рекомендацій ігор функціонує в умовах клієнт-серверної архітектури з розгортанням на хмарних сервісах. Система включає фронтенд-застосунок,

реалізований на базі React, що забезпечує інтерфейс користувача для взаємодії з платформою, перегляду списків ігор, рецензій, виконання пошукових запитів і подання форм.

На серверному боці розгорнуто модулі обробки запитів, створені з використанням Python та фреймворку FastAPI, що забезпечують побудову REST API для взаємодії з базами даних. Як основу для зберігання даних про ігри, жанри, сеттінги та пов’язані характеристики використано графову базу Neo4j, що дозволяє моделювати складні зв’язки у вигляді вузлів і ребер. Додатково, для зберігання тегів, запитів користувачів і записів з форм — застосовується реляційна база PostgreSQL, з якою інтеграція здійснюється за допомогою бібліотеки psycopg2.

Доступ користувачів до системи здійснюється через браузер із будь-якого пристрою, підключеного до мережі Інтернет. Завдяки використанню хмарних хостинг-провайдерів (наприклад, Vercel — для фронтенду, Render або Railway — для бекенду) досягається висока доступність і масштабованість рішення.

Централізована обробка запитів, модульна побудова архітектури, використання REST API та поділ даних між графовою й реляційною БД забезпечують гнучкість і можливість подальшого розширення функціональності системи.

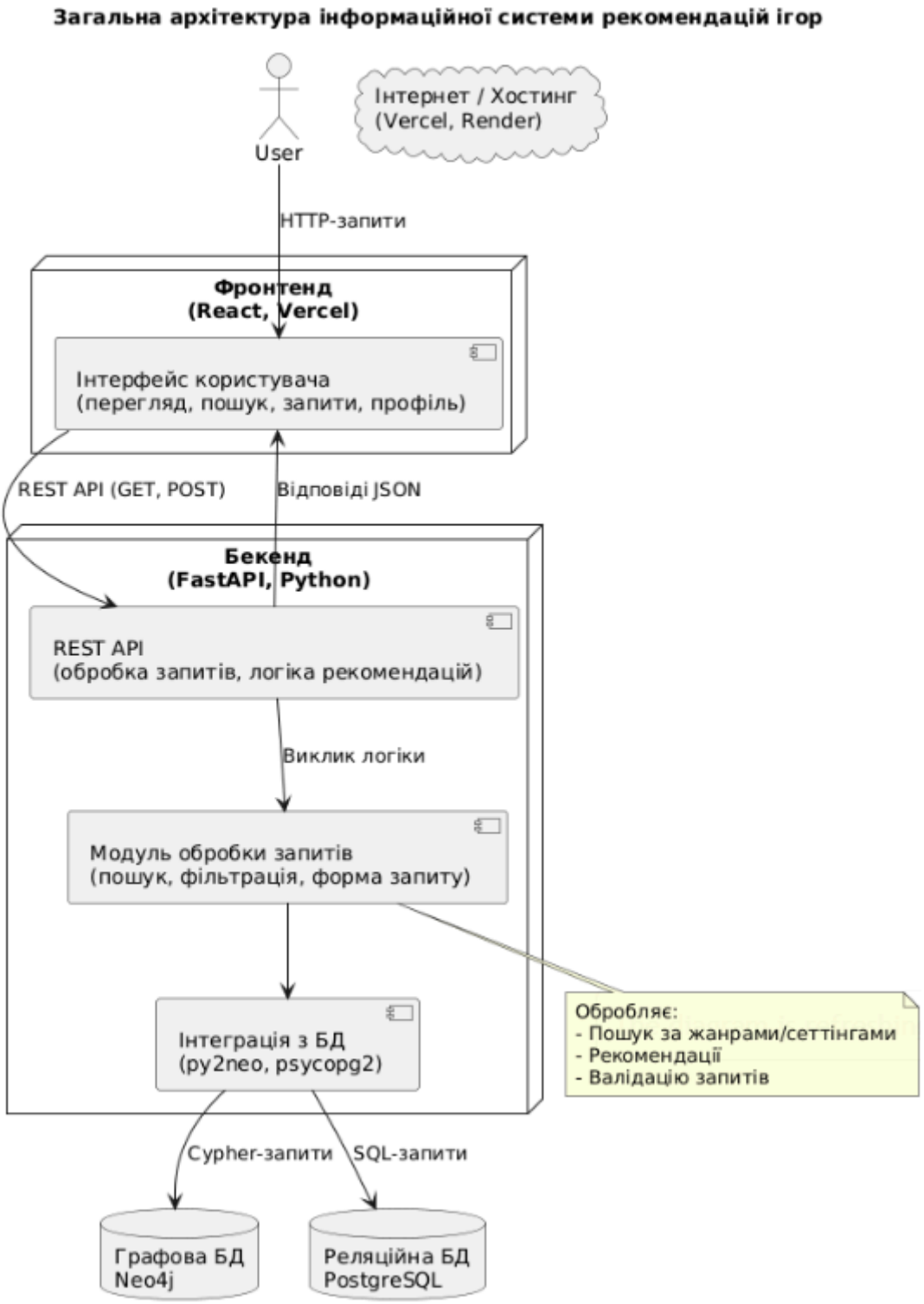

Рисунок 3.1 – Загальна схема автоматизації

3.2.2 Структура комплексу технічних засобів

Інформаційна підсистема рекомендаційної системи ігрової платформи функціонує в межах клієнт-серверної архітектури з підтримкою взаємодії

користувача через вебінтерфейс. Архітектура системи передбачає:

- використання віддаленого серверного середовища (наприклад, Render, Railway або інші хмарні провайдери), де розгорнуті бази даних (PostgreSQL для тегів, Neo4j для графових зв’язків) та бекенд-сервер на Python з мінімальними системними вимогами (1 CPU core, 512 МБ RAM, 1 ГБ SSD);
- доступ клієнтів до платформи через браузер, що дозволяє переглядати рекомендовані ігри, здійснювати пошук за жанром або сеттінгом, подавати запити на додавання нових ігор;
- використання відкритих або безкоштовних платформ для розгортання фронтенд-частини, таких як GitHub Pages, Vercel або Netlify, що дозволяє швидко та економно презентувати інтерфейс користувача.

Уся кодова база — як frontend, так і backend — зберігається у відкритому репозиторії на платформі GitHub, що забезпечує контроль версій, зручність колективної роботи та можливість інтеграції CI/CD процесів.

Інформаційна взаємодія реалізується через:

- REST API-запити з frontend до backend;
- запити з Python до Neo4j (через neo4j-driver) для отримання ігрових зв’язків;
- запити з Python до PostgreSQL (через psycopg2) для отримання або збереження інформації про теги, жанри та запити користувачів.

Завдяки такій інфраструктурі система є гнучкою, масштабованою та зручною для користувачів і розробників, а також може бути легко інтегрована в сторонні платформи цифрової дистрибуції або розгорнута в освітньому чи комерційному середовищі.

3.2.3 Опис автоматизованого робочого місця

Для взаємодії з системою користувачеві достатньо мати пристрій із

доступом до Інтернету та сучасним браузером. Типове АРМ:

Операційна система: Windows 10/11, Linux, macOS;
Браузер: Google Chrome (рекомендований), Firefox,

OperaGX; Процесор: Intel Core i3 або аналог;

ОЗП: 4 ГБ;

Дисковий простір: 500 МБ;

Засоби введення/виведення: клавіатура, миша/тачпад,

монітор; Засоби підключення: WiFi або Ethernet.

Для розробника:

Інструменти: Visual Studio Code, Node.js, Git, Python, Psycopg2

3.2.4 Схема мережі передачі даних

Система функціонує в умовах глобальної мережі. Доступ до системи забезпечується через Інтернет-з’єднання.

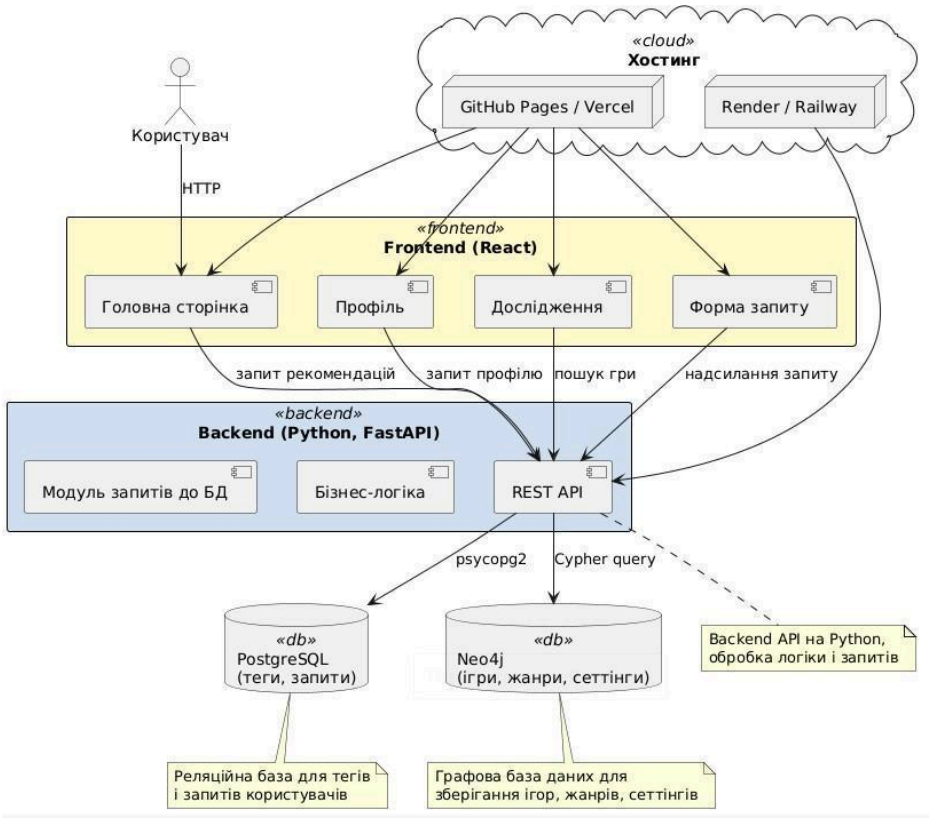

Рисунок 3.2 Схема роботи системи

Вибір засобів передачі:

- Підключення: WiFi / Ethernet / мобільний Інтернет;
- Протокол: HTTPS;
- Хостинг: Vercel / GitHub Pages для frontend; Render / Railway для backend;
- Канал зв’язку: мінімум 10 Мбіт/с

  (рекомендовано). Надійність і

  підтримка:
- гарантія від провайдера хостингу (uptime 99.9%);
- мінімальне споживання ресурсів;
- автоматичне масштабування у разі підвищеного навантаження;

3.3 Програмне забезпечення

Під програмним забезпеченням (ПЗ) інформаційних систем розуміється сукупність різних за функціями, взаємозалежних програмних і документальних засобів, що забезпечують створення та експлуатацію систем обробки даних. У контексті системи захищеного завантаження файлів, ПЗ є ключовим елементом, що реалізує всю бізнес-логіку, взаємодію з користувачами та управління даними.

3.3.1 Структура програмного забезпечення

Структура програмного забезпечення системи захищеного завантаження файлів є багатошаровою та включає базове (системне) ПЗ, прикладне (спеціалізоване) ПЗ та програмну документацію. Ця організація забезпечує ефективне функціонування системи та її подальшу підтримку.

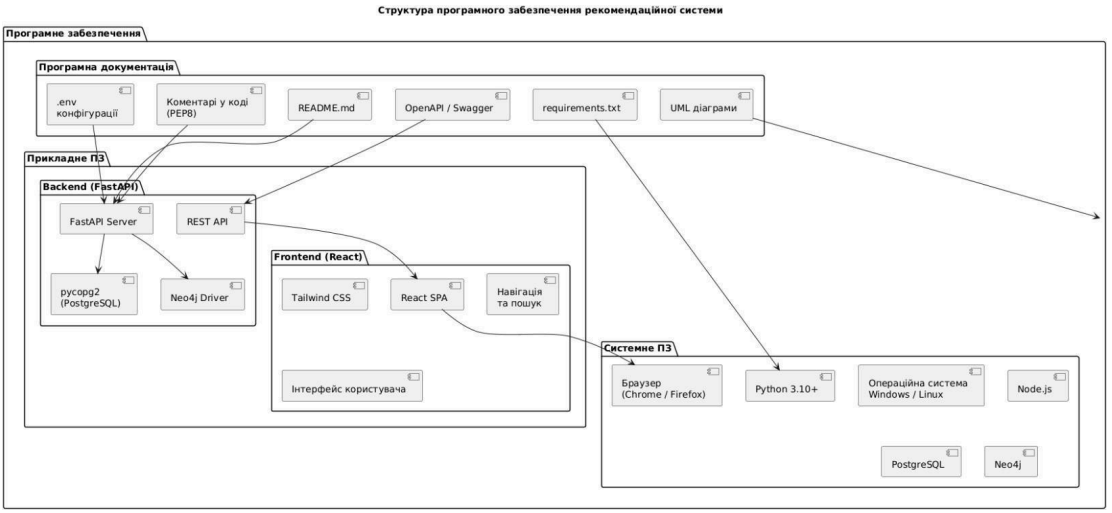

Рисунок 3.3 – Загальна структура програмного забезпечення

Структура ПЗ включає:

- Базове (системне) ПЗ: браузер, операційна система, Node.js,Python 3.10+; Прикладне ПЗ: frontend (React SPA), backend (сервер на Express.js), супровідні бібліотеки;
- Програмна документація: README, коментарі в коді, специфікації API, інструкції зі зборки та розгортання.

3.3.2 Системне програмне забезпечення

У рамках реалізації системи використовувалося таке системне ПЗ:
Операційна система: Windows 10+ (на боці розробника), Ubuntu Server 22.04 (на боці backend-хостингу);
Браузери (клієнт): Google Chrome, Mozilla Firefox,
OperaGX; Платформа розробки: Python 3.10+;
Пакетний менеджер: pip;
Серверна частина: FastAPI (для REST API);
СУБД: PostgreSQL (для реляційних даних про теги та запити), Neo4j (для графових зв'язків між іграми, жанрами, сеттінгами);

Бібліотека для взаємодії з PostgreSQL:
psycopg2; Бібліотека для взаємодії з
Neo4j: neo4j-driver;
Інструмент візуального моделювання: draw.io (для створення
UML-діаграм); Хостинг: Render (backend), Vercel або Netlify (frontend);
Системи контролю версій: Git + GitHub.

Обраний стек програмного забезпечення є відкритим, активно підтримується спільнотою та ідеально підходить для створення гібридної системи з високою продуктивністю та масштабованістю.

3.3.3 Прикладне програмне забезпечення

До прикладного ПЗ входять:
React — побудова інтерфейсу
користувача; Tailwind CSS — адаптивна
верстка;
React Router — реалізація SPA-навігації;
axios — виконання HTTP-запитів до REST API;
Formik + Yup — обробка форм (запит на нову гру) та
валідація. До серверного ПЗ входять:
FastAPI — створення API та обробка запитів;
pycopg2 — з’єднання з PostgreSQL та обробка
SQL-запитів; neo4j-driver — взаємодія з
графовою базою Neo4j;
uvicorn — ASGI-сервер для запуску FastAPI.

Функціональні можливості прикладного ПЗ охоплюють авторизацію/ідентифікацію користувачів, пошук ігор за жанрами та

сеттінгами, формування персоналізованих рекомендацій, надсилання та обробку запитів на нові ігри, а також взаємодію з базами даних у гібридному режимі.

3.3.4 Програмна документація

До складу програмної документації інформаційної підсистеми входять матеріали, які забезпечують розуміння принципів роботи системи, її функціональності, а також порядок встановлення, налаштування та експлуатації.

Основна документація міститься у репозиторії GitHub, де розміщено файл README.md, що містить загальний опис можливостей застосунку, інструкції зі встановлення необхідних залежностей, запуску frontend та backend частин, а також інформацію про системні вимоги для розгортання (Python 3.10+, Node.js 18+, PostgreSQL 15+, Neo4j 5.0+). Також у цьому файлі наведено приклади запитів до API та описано формат відповіді.

У межах роботи підготовлено технічний формуляр, що стисло подає архітектуру розробленої системи, описує взаємодію між компонентами, схему баз даних та API. Документація REST API автоматично згенерована засобами FastAPI (Swagger UI/OpenAPI) та доступна за внутрішнім маршрутом /docs.

Для кращого орієнтування в кодовій базі надано структуру основних каталогів і файлів backend та frontend частин. Зокрема, каталог /app/routers містить файли з описом маршрутів API, /app/db — модулі з’єднання з базами даних PostgreSQL і Neo4j,/frontend/src/components — окремі UI-компоненти інтерфейсу користувача, а /frontend/src/pages — логіку сторінок (Головна, Профіль, Дослідження).

Коментарі в коді пояснюють функціональність ключових функцій, використання параметрів, структуру відповідей та механізми обробки винятків. Це значно спрощує супровід проєкту та забезпечує можливість швидкої інтеграції нових розробників у процес.

Таким чином, програмна документація розробленої системи забезпечує повний цикл супроводу: від ознайомлення до розгортання і подальшого використання системи, як з боку технічних фахівців, так і кінцевих користувачів.

Нижче наведено рисунки з діаграмами (прототипами) інтерфейсної реалізації.

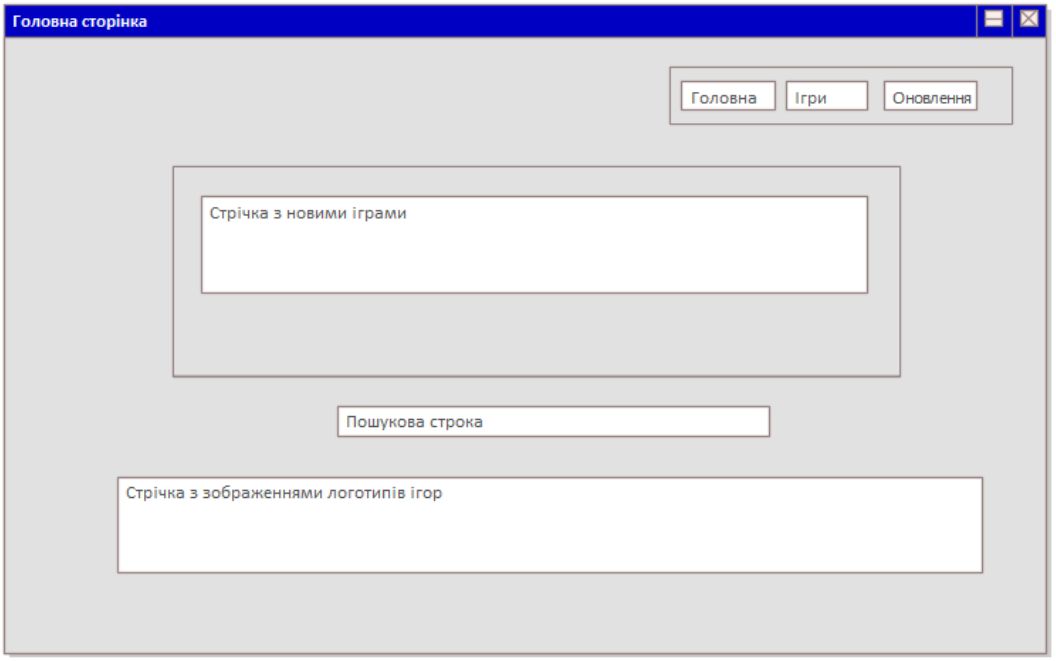

Рисунок 3.4 – Діаграма інтерфейсу Головна сторінка

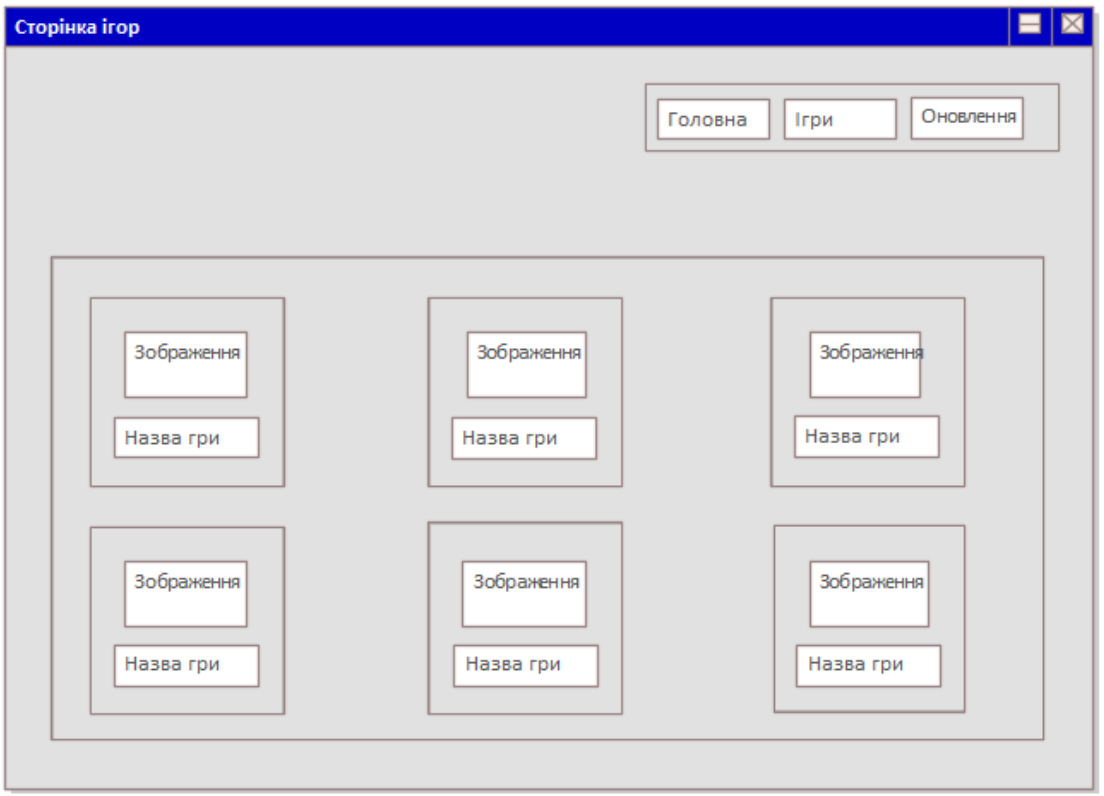

Рисунок 3.5 – Діаграма інтерфейсу Сторінка проектів

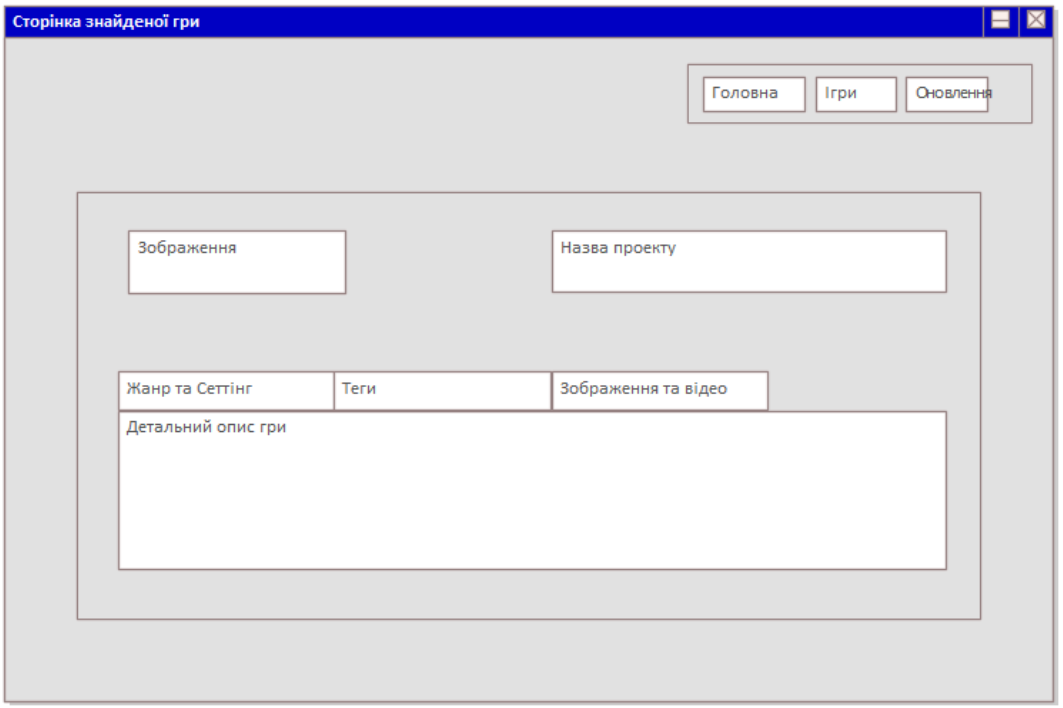

Рисунок 3.6 – Діаграма інтерфейсу Сторінка знайденої гри

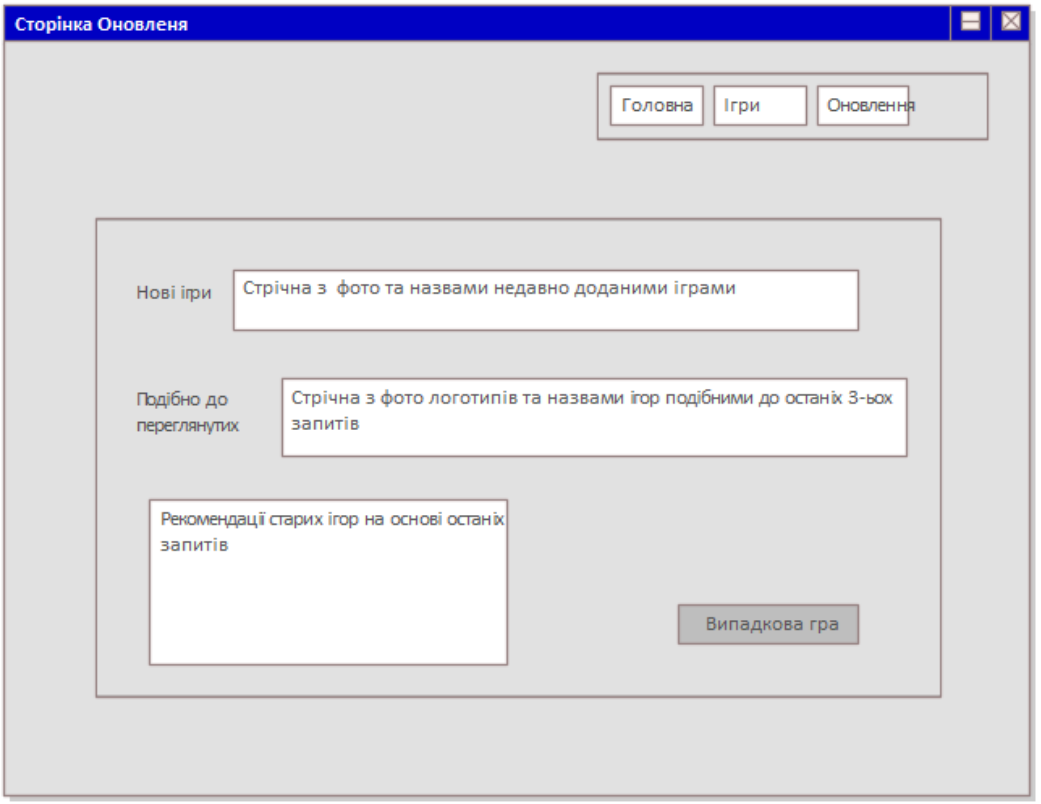

Рисунок 3.7 – Діаграма інтерфейсу Сторінка Оновлення

3.4 Результати реалізації рекомендаційної системи

У ході виконання бакалаврської роботи було реалізовано функціональний прототип рекомендаційної системи для ігрової платформи з використанням сучасних вебтехнологій та гібридного підходу до зберігання даних. Архітектура системи поєднує можливості графових баз даних (Neo4j) та реляційних (PostgreSQL), що дозволило створити адаптивну, масштабовану і прозору систему рекомендацій.

Серверна частина реалізована на основі Python-фреймворку FastAPI з використанням бібліотек py2neo та psycopg2 для взаємодії з Neo4j і PostgreSQL відповідно. Розроблений API обробляє запити користувачів, здійснює фільтрацію ігор за жанром і сеттінгом через графову модель, а також уточнює результати за тегами з урахуванням умов застосування кожного тегу, що зберігається у реляційній СКБД.

Однією з особливостей реалізації є інтеграція двох типів СКБД без

використання сторонніх сервісів, що дозволило забезпечити контроль над логікою формування рекомендацій та пояснюваність результатів. Завдяки цьому система може надавати обґрунтовані пропозиції на основі подібності ігор, спільних жанрів, сеттінгів і релевантних тегів.

Пілотне впровадження і тестування

Прототип було розгорнуто у хмарному середовищі Render з прив’язкою до репозиторію GitHub. Бази даних (Neo4j та PostgreSQL) були налаштовані як окремі контейнери, до яких здійснювався доступ через FastAPI. Тестування проводилось за наступними сценаріями:

Контрольний приклад:

Користувач сформував запит на гру у жанрі "Action" з тегом "multiplayer" та сеттінгом "sci-fi". Після надсилання запиту, система обробила його за кілька секунд, відібрала відповідні ігри з Neo4j і відфільтрувала результати через PostgreSQL. Відповідь містила перелік рекомендованих ігор із зазначенням, за якими критеріями вони були обрані.

Система стабільно працює у браузерах Chrome, Firefox та Edge, адаптується до роздільної здатності від 320px до 1920px, а логіка рекомендацій залишається прозорою та передбачуваною. Надсилання некоректних запитів (наприклад, відсутність жанру або неправильний формат email) коректно обробляється з виведенням повідомлення про помилку.

На відміну від типової реалізації, що базується на колаборативній фільтрації чи шаблонних системах, даний проєкт вирізняється пояснюваністю та можливістю контролю логіки формування рекомендацій. Гібридна архітектура дозволяє з легкістю масштабувати функціонал, розширити базу тегів або додати нові параметри (наприклад, рейтинги чи платформи).

Перспективи удосконалення

У майбутньому систему можна вдосконалити, через інтеграцію моделей поведінкових рекомендацій (наприклад, на основі оцінок чи історії перегляду). Також можливе впровадження системи оцінки точності (наприклад,

precision/recall),. Крім того, розглядається інтеграція з Telegram-ботом для миттєвих сповіщень про нові повідомлення з сайту. Додатково можливо розширення клієнтської частини з підтримкою реєстрації, особистого кабінету та історії запитів та інтеграція з зовнішніми платформами через API (Steam, GOG тощо).

Готовність до реального використання

Реалізована інформаційна система є завершеним функціональним прототипом, що може бути використаний як:

- демонстраційний приклад для освітніх цілей;
- основа для розробки стартап-платформи рекомендацій ігор;
- частина комплексної системи цифрової дистрибуції.

Використані технології — React, FastAPI, Neo4j, PostgreSQL — підтверджують актуальність і професійний рівень реалізації, а структура системи дозволяє швидке масштабування і гнучку адаптацію до нових задач.

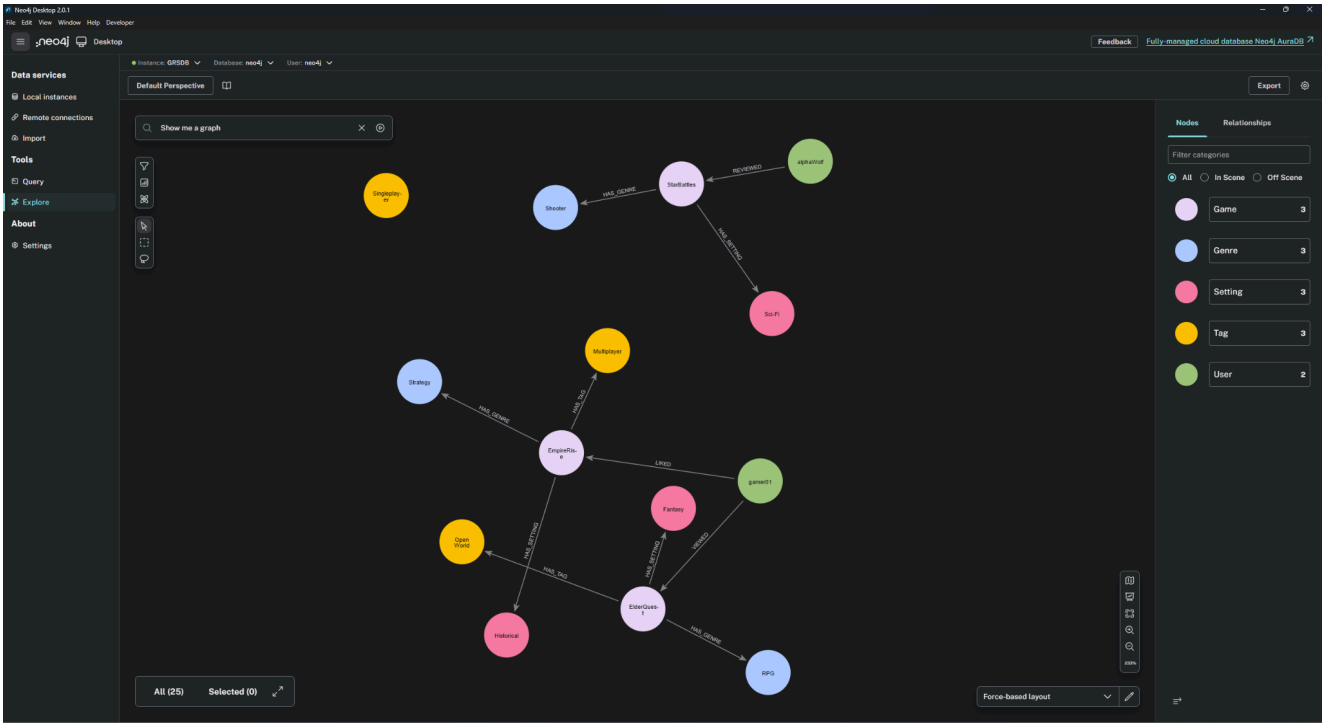

Рисунок 3.8 – Реалізація Neo4j Бази даних

Рисунок 3.9 – Загальний Figma прототип вебсторінки

ВИСНОВКИ

У процесі створення бакалаврського дипломного проєкту було досягнуто поставленої мети — реалізовано прототип сучасної, гнучкої та масштабованої рекомендаційної системи ігрової платформи з використанням графових і реляційних баз даних. Система дозволяє формувати персоналізовані добірки ігор на основі жанрово-сеттінгових зв’язків і тегової фільтрації, забезпечуючи користувача релевантними рекомендаціями з поясненням причин їх вибору.

На основі аналізу предметної галузі цифрової дистрибуції ігор було сформульовано вимоги до інформаційної підсистеми, визначено об’єкт і предмет дослідження, розроблено архітектуру системи, графову модель зв’язків у Neo4j та реляційну структуру тегів у PostgreSQL. Обрані технології — FastAPI, React, py2neo, psycopg2 — відповідають сучасним підходам до створення продуктивних і прозорих інформаційних систем.

Розроблена система має такі ключові особливості: ібридна структура з використанням Neo4j для моделювання взаємозв’язків між іграми, жанрами та сеттінгами, і PostgreSQL — для зберігання тегів та логіки фільтрації, зручною формою зворотного зв’язку з обробкою повідомлень на сервері, REST API, побудоване на FastAPI, клієнтський інтерфейс на React, який забезпечує адаптивну взаємодію з системою, включаючи фільтрацію за параметрами, перемикання теми та надсилання запитів на додавання нових ігор, а також підтримка збереження налаштувань користувача (наприклад, вибраної теми) та обробка форми запиту з перевіркою на сервері.

У рамках проєкту реалізовано інформаційне, технічне та програмне забезпечення, проведено тестування системи на основі контрольних прикладів, що включали різні варіанти запитів користувача, перевірку валідації введених даних, стійкість до помилок та адаптивність інтерфейсу до різних роздільностей екрана.

Розробка має високу практичну цінність, демонструє гнучкість і

потенціал для масштабування, а також може слугувати основою чи прикладом для створення інших рекомендаційних систем чи платформ цифрової дистрибуції.

Особистий внесок автора охоплює повний цикл розробки — від аналізу аналогів, постановки задачі та проєктування архітектури до створення моделей, реалізації серверної та клієнтської частини, налаштування зв’язків між базами даних, обробки запитів, тестування та документування проєкту.

У перспективі система може бути вдосконалена додавання аналітики взаємодії користувачів, інтеграція з зовнішніми сервісами (Steam API, Telegram-боти), розширення інтерфейсу (реєстрація, збереження історії запитів, мультимовність, сортування за рейтингами) і використання графових алгоритмів подібності (наприклад, Node Similarity, Personalized PageRank).

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wang S. et al. Graph Learning based Recommender Systems: A Review. arXiv preprint arXiv:2105.06339 (2021). https://arxiv.org/abs/2105.06339
2. Schlichtkrull M. et al. Modeling Relational Data with Graph Convolutional Networks. In ESWC 2018. https://arxiv.org/abs/1703.06103
3. Kim et al. (2021) — Sequential Recommendations on Board-Game Platforms. https://www.mdpi.com/2073-8994/12/2/210
4. Chen et al. (2021) — Sequential Recommendation in Online Games with Multiple Sequences, Tasks and User Levels . https://arxiv.org/pdf/2102.06950
5. Sutton C., Hall M. Data and Reality: A Timeless Perspective on Perceiving and Managing Information. Technics Publications, 2022.
6. Yang et al. (2022) — Large-scale Personalized Video Game Recommendation via Social-aware Contextualized Graph Neural Network (SCGRec). https://arxiv.org/pdf/2202.03392
7. Dinhani (2016) — Game Discovery: A Recommendation Algorithm for Video Games (Neo4j Community Blog).
8. https://neo4j.com/blog/video-game-discovery-recommendation-algorithm/
9. React documentation. – [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/getting-started.html
10. GitHub Documentation. Getting started with Git and GitHub. https://docs.github.com/
11. Vercel Documentation. Deployment of React apps. https://vercel.com/docs
12. Tailwind CSS Documentation. https://tailwindcss.com/docs
13. Node.js Documentation. https://nodejs.org/en/docs/

14. Express.js Guide. https://expressjs.com/
15. Bootstrap Documentation. https://getbootstrap.com/docs/
16. Netlify Documentation. https://docs.netlify.com/
17. Render Documentation. https://render.com/docs
18. Railway Documentation. https://docs.railway.app/
19. Psycopg2 Manual. https://pypi.org/project/psycopg2/
20. Psycopg2 Documentation. https://www.psycopg.org/docs/
21. PostgreSQL DB Manual. https://www.postgresql.org/docs/
22. Form Validation in React. https://react-hook-form.com/
23. Cypher Documentation. https://neo4j.com/docs/cypher/
24. Import data from a relational database into Neo4j. https://neo4j.com/docs/getting-started/appendix/tutorials/guide-import-relational-and-etl/
25. Uvicorn Documentation. https://www.uvicorn.org
26. Create React App Documentation. https://create-react-app.dev/docs/getting-started/
27. GitHub Pages Documentation. https://pages.github.com/
28. Internet filter. https://en.wikipedia.org/wiki/Internet_filter
29. Collaborative_filtering. https://en.wikipedia.org/wiki/Collaborative_filtering

ДОДАТКИ

Додаток А

```sql
CREATE TABLE tags (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100),
  description TEXT
);

CREATE TABLE genre_tag (
  genre_id INTEGER,
  tag_id INTEGER
);

CREATE TABLE setting_tag (
  setting_id INTEGER,
  tag_id INTEGER
);

CREATE TABLE game_tag (
  game_id INTEGER,
  tag_id INTEGER
);
```

Рисунок А.1 – Код бази даних SQL

```
// Додати жанри
CREATE (:Genre {name: "RPG"}),
       (:Genre {name: "Strategy"}),
       (:Genre {name: "Shooter"});

// Додати сеттінги
CREATE (:Setting {name: "Fantasy"}),
       (:Setting {name: "Sci-Fi"}),
       (:Setting {name: "Historical"});

// Додати теги
CREATE (:Tag {name: "Multiplayer"}),
       (:Tag {name: "Open World"}),
       (:Tag {name: "Singleplayer"});

// Додати ігри
CREATE (:Game {title: "ElderQuest", year: 2021}),
       (:Game {title: "StarBattles", year: 2022}),
       (:Game {title: "EmpireRise", year: 2020});

// Додати користувачів
CREATE (:User {username: "gamer01", age: 23}),
       (:User {username: "alphaWolf", age: 30});
```

Рисунок А.2 – Код вузлів Neo4j бази даних

```
// Жанри до ігор
MATCH (g:Game {title: "ElderQuest"}), (genre:Genre {name: "RPG"})
CREATE (g)-[:HAS_GENRE]->(genre);

MATCH (g:Game {title: "StarBattles"}), (genre:Genre {name: "Shooter"})
CREATE (g)-[:HAS_GENRE]->(genre);

MATCH (g:Game {title: "EmpireRise"}), (genre:Genre {name: "Strategy"})
CREATE (g)-[:HAS_GENRE]->(genre);

// Сеттінги до ігор
MATCH (g:Game {title: "ElderQuest"}), (s:Setting {name: "Fantasy"})
CREATE (g)-[:HAS_SETTING]->(s);

MATCH (g:Game {title: "StarBattles"}), (s:Setting {name: "Sci-Fi"})
CREATE (g)-[:HAS_SETTING]->(s);

MATCH (g:Game {title: "EmpireRise"}), (s:Setting {name: "Historical"})
CREATE (g)-[:HAS_SETTING]->(s);

// Теги до ігор
MATCH (g:Game {title: "ElderQuest"}), (t:Tag {name: "Open World"})
CREATE (g)-[:HAS_TAG]->(t);

MATCH (g:Game {title: "EmpireRise"}), (t:Tag {name: "Multiplayer"})
CREATE (g)-[:HAS_TAG]->(t);

// Користувачі — переглянули
MATCH (u:User {username: "gamer01"}), (g:Game {title: "ElderQuest"})
CREATE (u)-[:VIEWED]->(g);

MATCH (u:User {username: "gamer01"}), (g:Game {title: "EmpireRise"})
CREATE (u)-[:LIKED]->(g);

// Користувач — додав рецензію
MATCH (u:User {username: "alphaWolf"}), (g:Game {title: "StarBattles"})
CREATE (u)-[:REVIEWED {rating: 4.5, comment: "Great graphics!"}]->(g);
```

Рисунок А.3 – Код зв’язків Neo4j бази даних

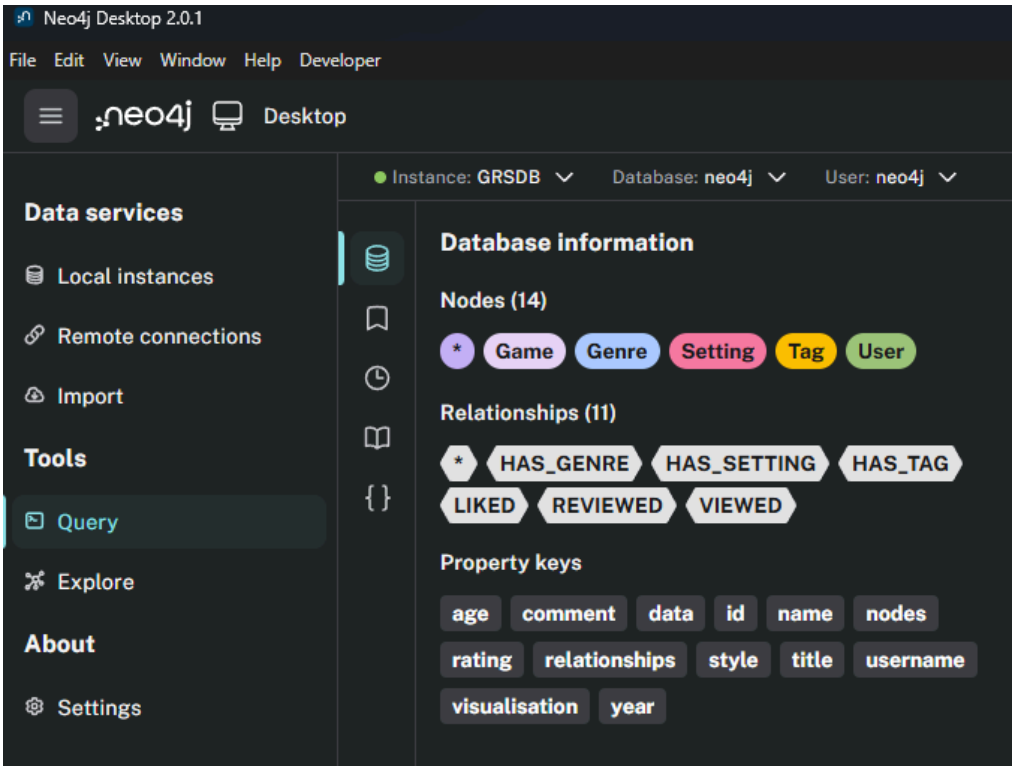

Рисунок А.4 – Neo4j база даних